FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Navegação e controlo de robôs móveis com atrelagem de reboques automática

Francisco Abílio Rodrigues Guerra Ferreira

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Professor Doutor António Paulo Gomes Mendes Moreira

Coorientador: Professor Doutor Germano Manuel Correia dos Santos Veiga

13 de Julho de 2017

Resumo

Com o propósito de utilizar robôs em tarefas de atracagem, pressupõe-se que estas sejam executadas de forma precisa e exata. No decorrer desse processo, podem-se salientar dois módulos ocorridos em simultâneo, nomeadamente a deteção e a aproximação do robô móvel ao objeto onde se pretende atracar um sistema do tipo reboque. Nessa perspetiva, esta dissertação tem como principal foco integrar e desenvolver um sistema que permita ao robô atracar de forma eficaz e segura. Desta forma, serão adaptados algoritmos de deteção e localização de marcos/balizas. Em simultâneo, serão analisadas as trajetórias possíveis de serem realizadas, tornando o robô capaz de se aproximar ao objeto.

Para o sistema de deteção da *docking station* será desenvolvido um algoritmo capaz de detetar uma linha. Do mesmo modo, serão testados dois métodos já implementados: o algoritmo de localização de *beacons* e o algoritmo *Perfect Match* e, numa fase posterior, serão ajustados para o caso de estudo em questão. Para o caso da aproximação do robô à *docking station*, este terá que percorrer uma trajetória retilínea recorrendo a um controlador seguidor de linha. As experiências foram, numa primeira fase testadas em ambiente de simulação sendo que, após a sua validação, foram testadas em robôs reais.

Abstract

With the objective to use robots executing docking tasks, those have to perform them with high precision and accuracy. In these tasks, we can point up two modules happening in simultaneous. In fact, one is responsible for the detection and the other one for guiding the robot towards an object where is supposed to dock. With this in mind, the main objective of this thesis is to develop a system that allows a robot to dock with precision and accuracy. Therefore, we will adapt algorithms for detection and localisation of landmarks. Simultaneously, we will analyse possible trajectories, which allows the robot to approach the object where is supposed to dock.

For the detection module, we will implement an algorithm to detect a line. We will also test two algorithms already implemented: a beacon-based localisation algorithm and the algorithm Perfect Match. We will adjust these algorithms for the main purpose of this dissertation. For approaching the dock, the robot will perform a linear trajectory using a Follow-Line controller. To begin, these experiments will be done in a simulation environment, then, after validation, they will be tested on a real robot.

Agradecimentos

Queria agradecer ao meu orientador Doutor Professor António Paulo Moreira e ao meu coorientador Doutor Professor Germano Veiga pelo acompanhamento durante esta dissertação assim como pela contribuição e apoio fornecido.

Gostaria de agradecer ao Héber Sobreira, ao Ivo Sousa e à Cláudia Rocha por me terem acolhido na equipa e ajudado durante o decorrer desta dissertação, quer através da partilha de informação, quer através do auxílio prestado durante este período.

Queria agradecer à Ana Catarina Simões pelo companheirismo e boa disposição demonstrados ao longo desta dissertação, permitindo assim tornar mais fácil esta última etapa.

Um agradecimento especial à Catarina Pinto pelo impacto que teve durante este desafio, pelo companheirismo assim como pela paciência demonstrada.

Ao Vítor Pinto um enorme obrigado pelo tempo despendido e dedicação mostrada durante esta dissertação.

Gostaria de agradecer à equipa do INESC TEC pela ajuda durante este período.

Um enorme agradecimento à minha família pela paciência e apoio demonstrados durante toda a minha vida.

E por último, mas não menos importante, aos meus amigos, que me acompanharam nesta dura e longa viagem, António Moreira, Alice Dias, David de Sousa, Gonçalo Carvalho, Filipe Pestana, Joana Almeida, João Moura, Luís Melo, Miguel Valdrez, Paulo Marú, Pedro Rafael, Pedro Relvas e Ricardo Coimbra, um grande obrigado pelo apoio e momentos de diversão vividos durante este tempo.

Francisco Ferreira

"I see now that the circumstances of one's birth are irrelevant.
It is what you do with the gift of life that determines who you are."

by Takeshi Shudo, The Art of Pokemon, the Movie: Mewtwo Strikes Back!.

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

PM	Perfect Match
AGV	Automated Guided Vehicle
SLAM	Simultaneous localization and mapping
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
RPROP	Resilient Back-Propagation
ROS	Framework Robot Operating System

Capítulo 1

Introdução

Neste capítulo é feita uma introdução à dissertação: "Navegação e controlo de robôs móveis com atrelagem de reboques automática". Começa-se por descrever a contextualização do tema, na secção 1.1, seguida da apresentação dos motivos intrínsecos ao desenvolvimento da tese, na secção 1.2. Posteriormente, na secção 1.3, são descritos os principais objetivos. Por fim, apresenta-se a arquitetura deste documento, na secção 1.4.

1.1 Enquadramento

Com o avançar dos tempos, a ideia de desenvolver sistemas mais autónomos tem-se tornado num pensamento imperativo. Dado isto, o ser humano tem desenvolvido soluções tecnológicas por forma a convergir para este modo de pensar. Desta feita, tem levado a um aumento do interesse do uso de veículos autónomos em meios industriais para tarefas específicas, como por exemplo transferência de carga de um ponto para outro. Estas funções, ao serem resolvidas de forma robotizada, diminuem interferências humanas e consequentemente aumentam a sua eficiência e o seu desempenho. No entanto, com tais soluções, novos problemas surgem, nomeadamente, em termos de precisão na execução de tarefas. Com o melhoramento da capacidade de processamento, a implementação de certos algoritmos, que outrora eram considerados pesados computacionalmente, tornou-se viável, tal como as melhorias a nível de sensores têm permitido obter medidas mais precisas.

Com estas evoluções, situações antes consideradas inexequíveis ou só possíveis de serem realizáveis recorrendo ao esforço humano estão a ganhar uma nova dimensão.

1.2 Motivação

Como já referido, as evoluções tecnológicas têm permitido ao ser humano almejar um sonho por este idealizado, o de um mundo onde os robôs libertam o homem do trabalho repetitivo e/ou perigoso. Contudo, este desenvolvimento não pode ocorrer sem que as técnicas de localização

e planeamento de trajetórias sejam aperfeiçoados. De facto, o não aprimoramento destes procedimentos podem gerar desvios que, quando significativos, podem levar ao insucesso da tarefa pretendida.

Desta forma, a necessidade de desenvolver métodos robustos de navegação e localização de robôs móveis tem vindo a aumentar.

1.3 Objetivos

O projeto terá como base a navegação e localização de um robô móvel. Este terá como intuito determinar a presença de uma *docking station* nas suas imediações. Para tal, serão adaptados algoritmos utilizados para a localização relativa de objetos (marcos ou balizas) por parte do robô, assim como será implementado um algoritmo de deteção de contornos naturais. Concluído este passo, o veículo autónomo terá de estipular qual a trajetória mais eficiente que precisará de percorrer de forma a executar a tarefa desejada. Terminado todo o processo, o robô terá que fazer o trajeto definido sem que o desvio seja significativo.

1.4 Estrutura do documento

Esta dissertação está dividida em sete capítulos. No capítulo 2 será apresentada a revisão bibliográfica, que consiste num aprofundamento do que já foi realizado sobre o tema de docagem, assim como algoritmos de deteção de objetos e planeamento de trajetórias.

No capítulo 3 serão apresentados os requisitos que o sistema terá que cumprir, os ambientes onde foram efetuados os testes e os veículos móveis utilizados.

Uma descrição do controlador utilizado assim como a sua afinação serão mencionados no capítulo 4.

No capítulo 5 serão expostos os algoritmos utilizados para a deteção da *docking station*.

No capítulos 6 são descritas as experiências efetuadas nos robôs reais assim como será feita uma análise dos resultados obtidos.

Por fim, serão apresentadas as conclusões tiradas no fim desta dissertação, bem como as sugestões quanto ao trabalho futuro.

Capítulo 2

Revisão Bibliográfica

Para a execução dos diferentes tipos de tarefas como carregar baterias e atrelar carga, entre outras, os robôs móveis precisam de fazer manobras de docagem de forma precisa e robusta.

W. Wang *et al.* [1] recorrem a sensores ultrasom para efetuarem o processo de docagem, tentando atracar dois robôs, sendo que um deles contém um emissor e o outro dois receptores (figura 2.1). Um manipulador é utilizado de modo a facilitar a manobra de docagem, diminuindo o efeito de erros sistemáticos (figura 2.2).

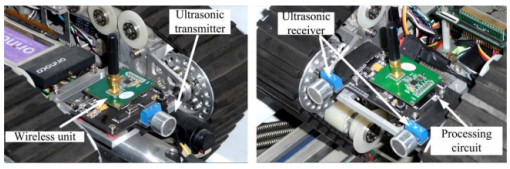


Figura 2.1: Colocação dos sensores ultrasom (Adaptado de [1]).

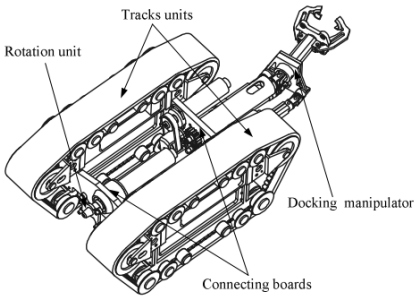


Figura 2.2: Robô com o manipulador (Adaptado de [1]).

Li Dazhai *et al.* [2] manuseiam o mesmo tipo de sensores na sua pesquisa, no entanto, apesar de utilizarem dois robôs, um deles possui um transmissor e o outro apenas um recetor (figura 2.3). Para facilitar a manobra, recorrem a uma estrutura cónica (figura 2.4).

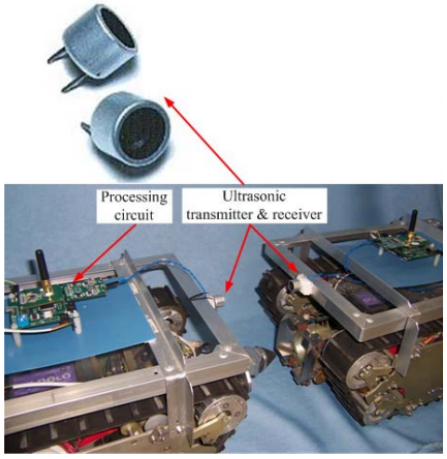


Figura 2.3: Colocação dos sensores ultra-som (Adaptado de [2]).

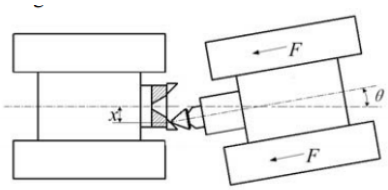


Figura 2.4: Vista de cima dos dois módulos a docarem (Adaptado de [2]).

R. C. Luo *et al.*[3] apresentam uma solução que utiliza dois pares de microfones para detetar a *docking station* (figura 2.5), aliados a processamento de imagem. Nesta experiência, a *docking station* emite sons que são detetados pelos microfones do veículo. Quando perto do objeto de interesse, recorrem a uma câmara para a aproximação final (figura 2.6).

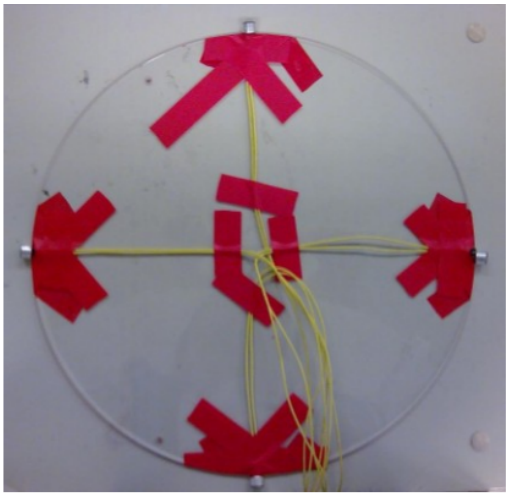


Figura 2.5: Pares de microfones utilizados (Adaptado de [3]).

Figura 2.6: Marco a detetar pela câmera (Adaptado de [3]).

Num outro artigo [4], os autores propõem uma solução que usa um laser *rangefinder* e processamento de imagem. Com os dados obtidos pelo laser detetam o ambiente à volta do robô e, através de uma câmara, detetam o marco artificial. Fundindo o resultado dos dois métodos de processamento, a posição da *docking station* é determinada (figura 2.7).

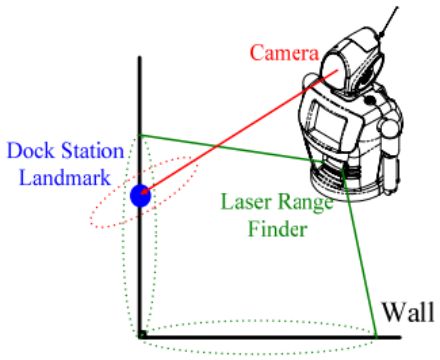


Figura 2.7: Exemplo da determinação da posição da *docking station* (Adaptado de [4])

M. Kim *et al.* [5] recorrem a um *transponder* de rádio frequências que se encontra na dock e a duas antenas no robô. Para a conseguir localizar, usam um método que estima a direção do sinal, onde utilizam a intensidade do sinal recebido.

D. Amarasinghe *et al.* [6], para determinarem a posição do objeto de interesse, recorrem a uma câmara. De igual modo, também é uma ferramenta utilizada por Zhang Zhigao *et al.* [7], por R. C. Luo *et al.* [8, 9] e por U. Kartoun *et al.* [10].

Desta forma, o processo de docagem pode ser dividido em duas fases. Uma primeira que trata da deteção da *docking station* e uma segunda de aproximação a esta. Para a deteção da doca será utilizado um laser *rangefinder*, visto que os sistemas de visão, em ambiente industrial, não são robustos, devido à variação das condições de iluminação.

2.1 Deteção da *docking station*

2.1.1 Métodos de ajuste de mapa

No âmbito da deteção e localização relativa de objetos (marcos ou balizas) por parte do robô, vários algoritmos têm emergido de forma a solucionar este problema. No domínio dos algoritmos de ajuste de mapa existem alguns que se destacam pela sua baixa necessidade de cálculo e simplicidade de utilização, nomeadamente o *Perfect Match*.

Perfect Match

Para que seja possível que um robô móvel se movimente sem ser necessária supervisão humana, este necessitará de determinar a sua localização. Desta forma, Martin Lauer *et al.* [11] desenvolveram o algoritmo *Perfect Match*. Este tem sido muito utilizado, em especial no *RoboCup MidSize League* (MSL). O algoritmo permite obter uma solução de forma rápida, na ordem dos 3 milissegundos, e, por não ser computacionalmente pesado, é possível utilizá-lo a frequências elevadas. Igualmente, este algoritmo tem a vantagem de não necessitar de marcos artificiais, sendo possível obter uma localização recorrendo só a características naturais do meio envolvente. Estas características são detetadas através de uma câmara. No artigo desenvolvido por Miguel

Pinto *et al.*[12] apresentam uma adaptação do algoritmo acima descrito de forma a ser aplicado em ambiente industrial. De facto, recorrem a um laser *rangefinder* para detetar as características do meio ambiente. Para além desta adaptação, aplicam algumas melhorias ao nível do filtro de Kalman, utilizado para estimar a pose do robô, entre outras. Este filtro permite fazer a fusão entre a parte sensorial e a odometria, obtendo uma estimativa da posição do robô.

Este algoritmo pode ser adaptado de forma a que seja possível determinar a localização de objetos (marcos) em relação ao robô.

2.1.2 Feature Matching

Existem vários tipos de sensores de que um robô móvel poderá tirar partido de modo a ser possível localizar-se. De facto, é possível utilizar um sensor laser *rangefinder*, sendo para tal necessário processar os dados obtidos por este. Desta forma, procura-se detetar características invulgares no ambiente envolvente, podendo estas ser naturais, ou seja elementos físicos que existam no meio, ou características artificiais, obtidas após a introdução de marcos artificiais ou balizas.

Por um lado, podem-se detetar cantos [13, 14, 15, 16], linhas [17, 18, 19, 20, 21] ou mesmo superfícies curvas [22, 23, 24, 25], descobrindo quais os dados que pertencem a estas assim como qual o centro e raio das mesmas. R. C. Luo *et al.* [26] apresentam um modo de localização de um veículo móvel utilizando um extrator de características naturais aliado ao um filtro de Kalman (EKF). Q. Xu *et al.*[27] apresentam um método que faz a fusão entre a parte sensorial com a odometria, usando um *Unscented Kalman Filter* (UKF).

Por outro lado, marcadores artificiais podem ser detetados de diferentes formas. Efetivamente, estes podem ser ativos, ou seja, emitem um sinal que será detetado pelo robô móvel, ou passivos, refletem o sinal emitido pelo veículo móvel. Existem marcadores ativos que emitem sinais Bluetooth, utilizados por F. Schwiegelshohn *et al.* [28], que por usa vez são usados para a localização e mapeamento (SLAM) do ambiente. Já os marcadores utilizados por J. Krejsa e S. Věchet [29] emitem sinais infravermelhos, permitindo assim localizar o robô móvel através de técnicas de triangulação. O mesmo tipo de marcadores é utilizado por A. Lobo *et al.* [30], que apresentam uma solução de localização recorrendo a técnicas de trilateração e com recurso a sensores *Dead Reckoning*. Com marcadores passivos, o veículo emite um sinal e espera o seu retorno, determinando assim a distância entre este e os *beacons*. Esta solução é utilizada por Lee Sooyong e Song Jae-Bok [31] e por J. López *et al.*[32] para localização do veículo móvel.

2.2 Planeamento de trajetória

Kuo-Lan Su Lin *et al.*[33] para conseguirem realizar a atracagem do veículo autónomo, este tem de iniciar um processo de detecção da *docking*, permitindo-lhe determinar qual o ângulo que ele faz com esta (θ), assim como a sua distância à mesma (R). Com estes dados, através de regras de trigonometria, é possível determinar qual a distância para o ponto mais próximo que pertença à reta que permita ao robô estar alinhado com a *docking station* (X) (Equação 2.1).

$$cos(\theta) = \frac{X}{R} \Leftrightarrow X = R \cdot cos(\theta) \tag{2.1}$$

Determinada esta posição, inicia-se o processo de atracagem. Este consiste em conduzi-lo até à localização previamente definida. Terminada esta tarefa, o veículo roda 90 graus, ficando direcionado para a estação de carregamento, como é possível observar nas imagens 2.8 e 2.9. Isto permite iniciar a etapa final, que se limita a aproximar o robô da *docking station*.

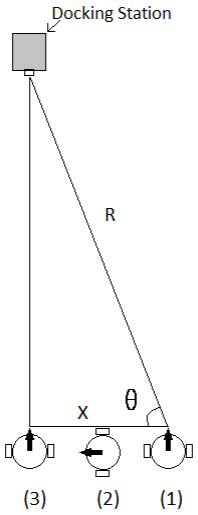


Figura 2.8: Exemplo quando a *docking station* se encontra à esquerda do robô. (1) pose inicial, (2) pose intermédia, (3) pose final já orientado para a *docking station*.

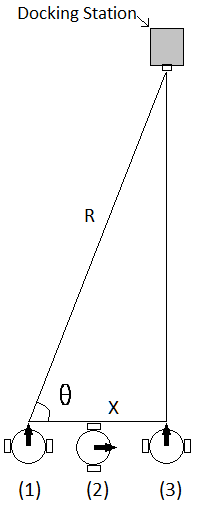


Figura 2.9: Exemplo quando a *docking station* se encontra à esquerda do robô. (1) pose inicial, (2) pose intermédia, (3) pose final já orientado para a *docking station*.

K. L. Su *et al.* [34] descrevem um método de processo de atracagem num sistema de múltiplas *docking stations*.

Quando o nível da bateria do robô atinge um valor abaixo de um certo limiar, este envia um sinal às estações. Estas respondem com o seu estado, sendo que o veículo móvel determina qual a que se encontra mais perto e livre. Identificada a *docking station*, o veículo autónomo inicia o processo de deteção do seu marco. Terminando esta etapa, o robô dirige-se para uma posição a 0.4 metros do objetivo final, como é possível observar na imagem 2.10. Para tal, planeia uma trajetória usando o algoritmo A^* (A-star). Concluído este passo, pode-se iniciar a fase final de dockagem.

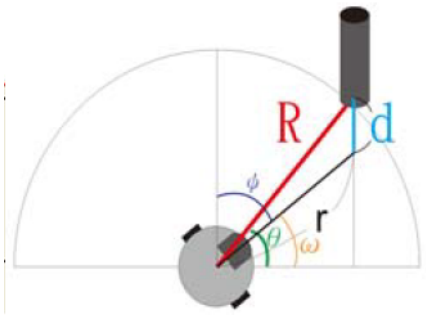


Figura 2.10: Exemplo do processo de dockagem (Adaptado de [34])

No projeto MIRA [35], é descrito um conjunto de procedimentos para o planeamento do processo de atracagem, usando dados fornecidos pelo laser.

Numa primeira fase, o *template* da *docking station*, a posição inicial e final do sistema de docagem são guardados. Quando o robô pretende docar, dirige-se para a posição inicial orientado para a doca, de onde se localiza em relação a esta, utilizando o *template* guardado. Terminada esta fase, o robô móvel dirige-se para a posição final.

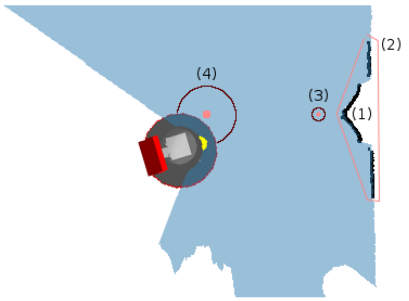


Figura 2.11: Exemplo do procedimento de dockagem (Adaptado de [35]). A pose final é descrita pelo ponto (3), sendo que a inicial é descrita pelo ponto (4). Os círculos descritos à volta destes pontos, representam a tolerância máxima permitida. O template para permitir determinar a localização do robô móvel é dado pelos pontos (1) e (2).

Capítulo 3

Ambiente de teste utilizado

Neste capítulo será efetuado um levantamento dos requisitos a que o sistema terá de corresponder, assim como descrever o ambiente de simulação e o ambiente real onde foram realizados testes de validação. No fim, serão apresentados os robôs móveis reais utilizados para essa mesma fase de testes.

3.1 Requisitos impostos ao sistema

Num processo de docagem, a precisão do sistema é crucial. Como tal, o sistema de deteção e aproximação à *docking station* necessita de ser exato e preciso.

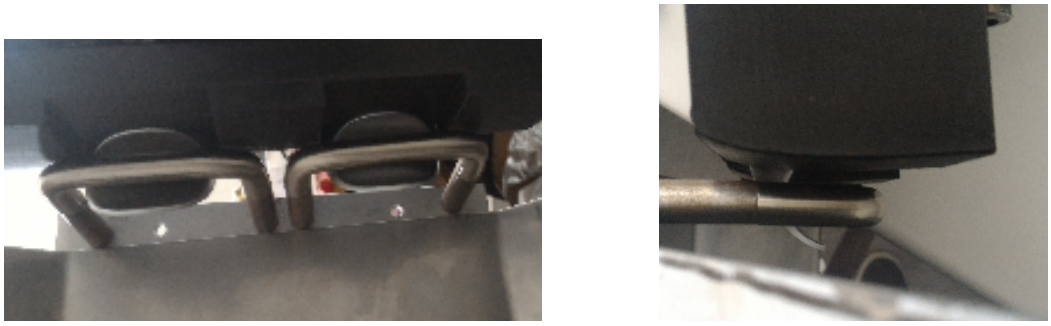
O sistema de atracagem parte do princípio de funcionamento de um reboque, utilizando duas estruturas em U na doca (Figura 3.1) e dois ganchos (Figura 3.2) que fazem a interface entre o braço localizado no robô móvel e as argolas em U.

Figura 3.1: Argolas em U onde se pretende docar.

(a) Vista de frente. (b) Vista de lado.

Figura 3.2: Ganchos utilizados para o processo de docagem.

As figuras 3.3a e 3.3b retratam uma situação de docagem bem sucedida, demonstrando a interação entre os componentes de atracagem.



(a) Vista de frente. (b) Vista de lado.

Figura 3.3: Situação bem sucedida de docagem.

Desta forma, foi determinado o erro máximo permitido longitudinalmente e transversalmente. Como é possível observar nas figuras 3.4 e 3.5, as argolas em U têm 62 milímetros de comprimento e 65 milímetros de largura, enquanto que os ganchos têm 38 milímetros de comprimento e 21 milímetros de largura. O gancho apresenta uma leve curvatura nas partes laterais, o que facilita o escorregamento do mesmo para o interior das estruturas ovais, atracando de forma bem sucedida.

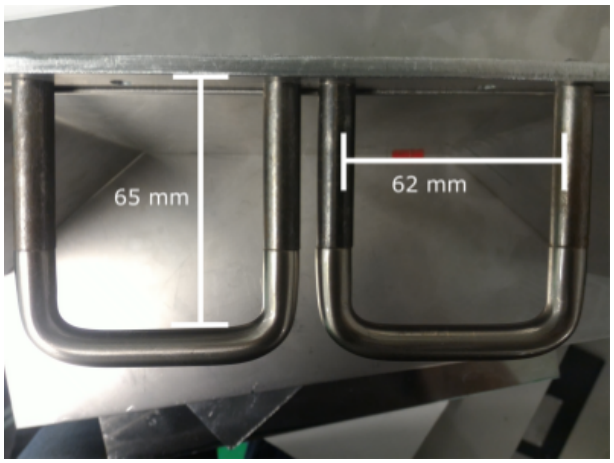
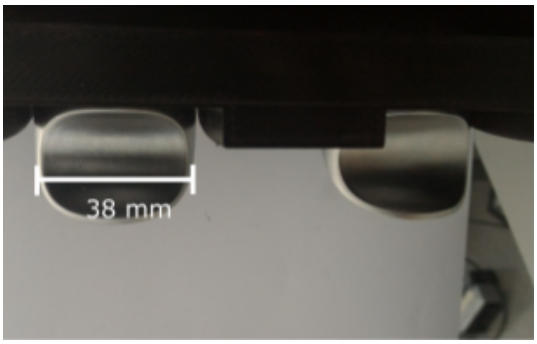
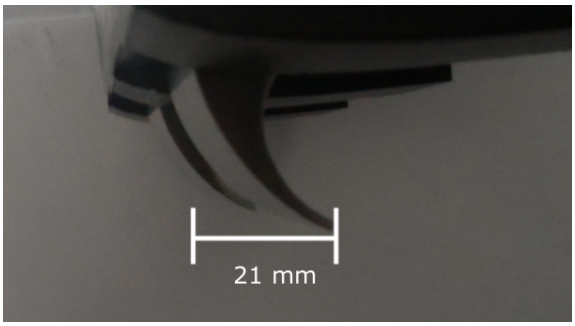


Figura 3.4: Argolas em U com medidas



(a) Vista de frente com medidas



(b) Vista de lado com medidas

Figura 3.5: Ganchos

Desta forma, estando os ganchos centrados com as argolas em U, o erro máximo permitido é de $\pm$ 12 milímetros transversalmente e de $\pm$ 21 milímetros longitudinalmente.

3.2 *Framework Robot Operating System* (ROS)

O sistema desenvolvido baseia-se na *framework* ROS, que é direcionada para o desenvolvimento de software para aplicações no âmbito da robótica.

O ROS inclui um amplo conjunto de ferramentas e bibliotecas que simplificam o desenvolvimento de sistemas robóticos. Este apresenta uma arquitetura modular que facilita o trabalho em equipa no desenvolvimento de soluções, como também na partilha de módulos. De facto, qualquer pessoa pode desenvolver o seu programa, que é facilmente integrado no sistema, visto que a framework é responsável pelos processos de comunicação entre todos os módulos da aplicação.

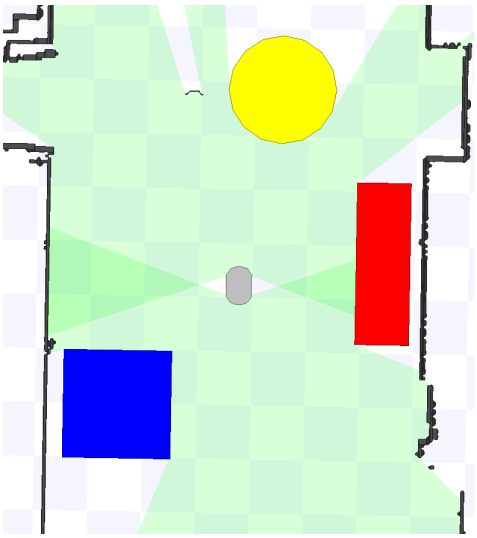
Figura 3.6: Logótipo da *framework* ROS.

3.3 Ambiente de simulação

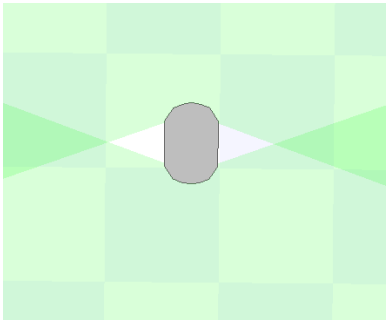
Antes do teste em ambiente real, foi efetuada uma validação do que foi implementado, testando-se no ambiente de simulação *Stage*. O simulador foi utilizado para validar o algoritmo de deteção de uma linha, tendo sido, inicialmente, testado sem ruído nas medidas do laser e, numa fase posterior, foi adicionado ruído gaussiano. Também foi usado para se testar os outros algoritmos de deteção da *docking station* : algoritmo de localização de *beacons* e algoritmo *Perfect Match*, bem como a manobra de aproximação escolhida ao objeto de interesse.

O *Stage* é um simulador 2D simplificado desenvolvido para sistemas operativos do tipo Linux, permite a interação com sistemas baseados na *framework* ROS e é possível programá-lo usando vários tipos de linguagem como C, C++, Ruby e Python, no entanto não apresenta motor de física.

Para o ambiente de simulação, utilizou-se um robô idêntico ao Edgar (apresentado na Secção 3.5.1), com o mesmo tipo de tração assim como o mesmo número de sensores (Figura 3.7b). Foram também colocados objetos aleatórios para simular o ambiente dinâmico em que o robô irá interagir (Figura 3.7a).



(a) Ambiente de simulação. Foram colocados objetos à volta do robô para simular um ambiente dinâmico.



(b) Robô móvel simulado. Região verde representa as medidas dos sensores laser.

Figura 3.7: Stage.

3.4 Ambiente real

Para verificação e validação do que se tinha implementado e testado no simulador, foi necessário experimentar em robôs reais.

Os testes feitos foram realizados na sala I-108 da Faculdade de Engenharia da Universidade do Porto. Esta sala apresenta uma ambiente dinâmico com a circulação de pessoas assim como colocação ou remoção de mobília ou de robôs móveis, como é possível observar nas figuras 3.8 e 3.9.

Figura 3.8: Fotografia tirada no início da fase de testes

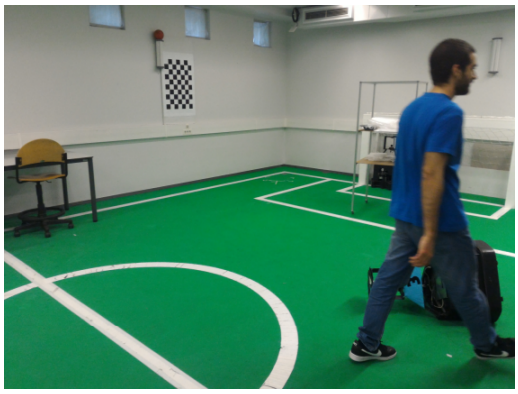


Figura 3.9: Fotografia tirada durante a fase de testes

3.5 Robôs móveis utilizados

Durante a fase de testes foram utilizados dois robôs móveis, apresentados nas figuras 3.11 e 3.10.

Figura 3.10: Edgar.

Figura 3.11: *Jarvis*.

3.5.1 Edgar

O *Edgar* é um veículo autónomo que possui dois lasers de segurança SICK S300. Um deles situa-se na parte frontal (Figura 3.12) sendo que o outro se situa na parte traseira (Figura 3.13), permitindo assim obter um campo de visão de 360 graus.

Figura 3.12: Laser frontal do Edgar.

Figura 3.13: Laser traseiro do Edgar

Apresenta um sistema de tração diferencial, com duas rodas motrizes localizadas no centro do robô e quatro rodas-loucas de forma a dar-lhe estabilidade.

3.5.1.1 Tração Diferencial

Esta é uma das configurações mais conhecidas pela sua simplicidade e baixo custo de implementação. Composta por duas rodas de tração, em que cada uma possuí um motor, e por uma ou mais rodas livres de forma a aumentar a estabilidade do robô. No entanto, esta configuração só permite ao veículo autónomo ter dois graus de liberdade. De facto, este só se consegue movimentar com velocidade linear e angular, não lhe permitindo deslocar-se num movimento lateral.

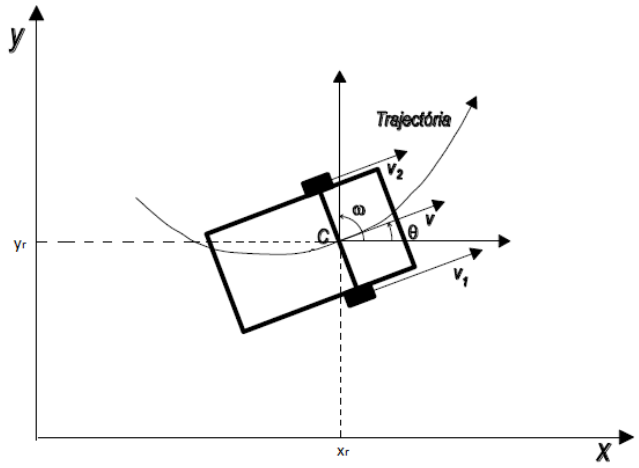


Figura 3.14: Tração Diferencial(Adaptado de [36])

Admitindo $\begin{bmatrix} x_r(t) \\ y_r(t) \\ \theta(t) \end{bmatrix}$ como sendo o estado do robô em relação ao referencial global

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x_r(t) \\ y_r(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} v(t)\cdot\cos(\theta(t)) \\ v(t)\cdot\sin(\theta(t)) \\ \omega(t) \end{bmatrix} \tag{3.1}$$

onde $v(t)$ representa a velocidade linear e $\omega(t)$ a velocidade angular, obtidas da seguinte forma

$$\begin{aligned} v(t) &= \tfrac{v_1(t)+v_2(t)}{2} \\ \omega(t) &= \tfrac{v_1(t)-v_2(t)}{b} \end{aligned} \tag{3.2}$$

sendo $v_1(t)$ a velocidade da roda 1, $v_2(t)$ a velocidade da roda 2 e b a distância entre rodas.

Integrando e passando para o tempo discreto a equação 3.2, é possível obter os deslocamentos linear e angular, como é possível observar na equação 3.3

$$\begin{aligned} d(k) &= \tfrac{d_1(k)+d_2(k)}{2} \\ \Delta\theta(k) &= \tfrac{d_1(k)-d_2(k)}{b} \end{aligned} \tag{3.3}$$

Desta forma, utilizando a aproximação em tempo discreto das diferenças centradas, obtém-se a estimativa da pose do robô no instante k+1 dada a pose no instante anterior e os deslocamentos efetuados:

$$\begin{cases} x_r(k+1) = x_r(k) + d(k) \cdot \cos\left(\theta(k) + \frac{\Delta\theta(k)}{2}\right) \\ y_r(k+1) = y_r(k) + d(k) \cdot \sin\left(\theta(k) + \frac{\Delta\theta(k)}{2}\right) \\ \theta(k+1) = \theta(k) + \Delta\theta(k) \end{cases} \tag{3.4}$$

3.5.2 *Jarvis*

O *Jarvis* é um veículo autónomo constituído por dois lasers, um de segurança SICK S3000 (Figura 3.15), montado a cerca de 0.2 m do solo, e um outro de navegação NAV350 (Figura 3.16), montado no topo na estrutura central a cerca de 2,0 m do solo.

Figura 3.15: Laser de segurança do *Jarvis*

Figura 3.16: Laser de navegação do *Jarvis*

O sistema de localização deste robô móvel é realizado através do laser SICK NAV350 usando 4 refletores cilíndricos com 9 cm de diâmetro (Figura 3.17), obtendo uma precisão até 4 milímetros[1].

[1] *https://www.sick.com/de/en/detection-and-ranging-solutions/2d-lidar-sensors/nav3xx/nav350-3232/p/p256041*

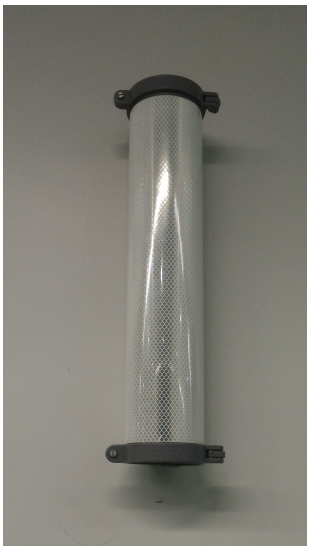


Figura 3.17: *Beacon* cilíndrico

Apresenta um sistema de tracção triciclo, com duas rodas traseiras e uma roda motriz na frente.

3.5.2.1 Tração Triciclo

Devido à sua robustez em relação a derrapagens, esta configuração é muito utilizada em ambientes industriais, em que a precisão na execução de tarefas é crucial.

Esta apresenta uma roda dianteira associada a um motor de tração e duas rodas traseiras para dar estabilidade ao veículo. O controlo de direção deste tipo de tração é feita pelo bloco da frente.

Para determinar as equações de cinemática, considera-se $\begin{bmatrix} x_r(t) \\ y_r(t) \\ \theta(t) \end{bmatrix}$ o estado do robô e pode-se utilizar a aproximação por "bicicleta"sem perda de generalidade

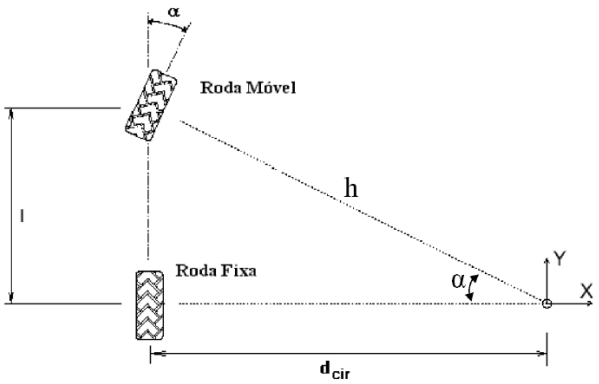


Figura 3.18: Aproximação considerada (Adaptado de [36])

Determinando o centro instantâneo de rotação (CIR)

$$\tan(\alpha) = \frac{l}{d_{cir}} \Leftrightarrow d_{cir} = \frac{l}{\tan(\alpha)} \tag{3.5}$$

e, sabendo que a velocidade linear da roda traseira é determinada pela equação

$$V_b = \omega \cdot d_{cir} \tag{3.6}$$

pode-se substituir a equação 3.6 pela equação 3.5, obtendo-se a seguinte equação

$$\omega = \frac{V_b \cdot \tan(\alpha)}{l} \tag{3.7}$$

Com a velocidade linear obtida na equação 3.6 e angular obtida na 3.7, é possível determinar a posição e a orientação do robô através da equação 3.4.

Capítulo 4

Controlador de trajetória

Neste capítulo, será explicado o controlador de trajetória utilizado , assim como a afinação do mesmo.

Como referido anteriormente, o processo de atracagem requer sistemas precisos e exatos. Desta forma, não só o sistema de localização do marcador tem que ser preciso, mas o mesmo tem de acontecer com o controlador de trajetória, que permitirá fazer a aproximação à *docking station*. Como tal, este terá de ser afinado de modo a obterem-se erros finais baixos.

4.1 Controlador seguidor de linha

Para a aproximação final, o robô móvel terá de percorrer uma reta que lhe permita aproximar do objeto onde tenciona atracar. Como tal, utilizou-se um controlador seguidor de linha, que pretende minimizar dois tipos de erro, um primeiro relativo à distância do robô em relação à reta e um segundo em relação à diferença entre a orientação do veículo móvel e a do segmento de reta (Figura 4.1).

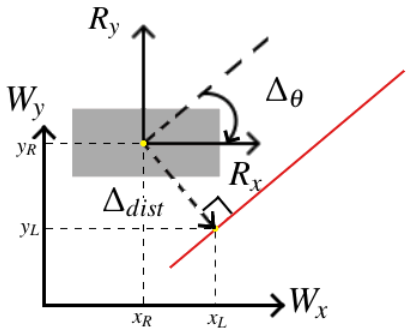


Figura 4.1: Erros associados ao controlador seguidor de linha (linha a vermelho). Δ_θ representa o erro de ângulo, Δ_{dist} o erro de distância e (R_x,R_y) o referencial relativo ao robô móvel. (x_R,y_R) representam as coordenadas do robô e (x_L,y_L) as coordenadas do ponto da linha mais próximo do robô, ambas em relação ao referencial global (W_x,W_y).

O controlador utilizado tem como entradas duas equações paramétricas que descrevem as coordenadas X e Y ao longo da trajetória. De facto, a linha é descrita da seguinte forma

$$\begin{aligned} X(u) &= A_x + B_x \cdot u \\ Y(u) &= A_y + B_y \cdot u \end{aligned}, u \in [0,1] \tag{4.1}$$

em que $(X(0),Y(0))$ representam as coordenadas do ponto inicial da reta e $(X(1),Y(1))$ as coordenadas do ponto final desta.

O erro de distância é determinado através do ponto (x_L,y_L) e das coordenadas do robô (x_R,y_R), utilizando a fórmula 4.2.

$$\Delta_{dist} = \pm\sqrt{(x_L - x_R)^2 + (y_L - y_R)^2} \tag{4.2}$$

As coordenadas (x_L,y_L) são determinadas recorrendo às equações paramétricas, de facto estas são percorridas de forma a determinar o ponto da linha mais próximo do robô (Equação 4.3).

$$\begin{aligned} &\underset{u}{\text{minimizar}} && \sqrt{(X(u) - x_R)^2 + (Y(u) - y_R)^2} \\ &\text{sujeito a} && u \in [0,1] \end{aligned} \tag{4.3}$$

O sinal da equação 4.2 é dado pela equação 4.4.

$$\begin{cases} 1 & if\ normalizeAngle(atan2(y_L - y_R, x_L - x_R) - \theta_R) \geq 0 \\ -1 & if\ normalizeAngle(atan2(y_L - y_R, x_L - x_R) - \theta_R) < 0 \end{cases} \tag{4.4}$$

onde θ_R representa o ângulo do referencial relativo do robô (R_x,R_y) em relação referencial global (W_x,W_y). A função $atan2$ representa a função arco tangente, sendo que retorna também o quadrante onde se situa o ângulo. A função $normalizeAngle()$ normaliza o ângulo resultante entre -180 graus e 180 graus.

O erro de ângulo é dado pela fórmula 4.5.

$$\Delta_\theta = normalizeAngle(\theta_R - \theta_L) \tag{4.5}$$

onde θ_R representa o ângulo do referencial relativo do robô (R_x,R_y) em relação referencial global (W_x,W_y) e θ_L a orientação da reta em relação ao mesmo referencial.

Este sistema de controlo é constituído por dois controladores com o objetivo de minimizar os erros descritos na figura 4.1, para tal, atua na velocidade de rotação do veículo móvel (Figura 4.2), sendo que a sua velocidade linear se mantém constante ao longo da trajetória. Os ganhos deste controlador serão determinados na subsecção 4.1.1.

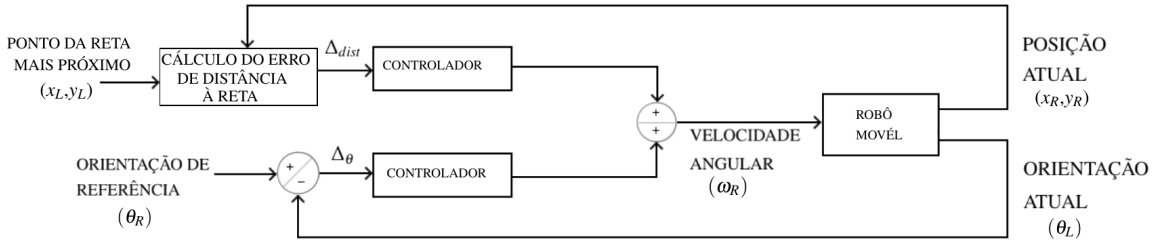


Figura 4.2: Exemplificação gráfica do controlador mencionado.

4.1.1 Afinação dos controladores

Numa primeira fase, verificou-se a resposta dos motores a comandos de velocidade linear e angular. A taxa de aquisição das velocidades do motores foi feita a 50 milissegundos.

Para verificar a resposta a comandos de velocidade linear, aplicou-se uma onda quadrada com um valor mínimo de 0.05 m/s e um valor máximo de 0.4 m/s (Figura 4.3). Estes valores foram escolhidos de forma a evitar a zona morta do motor, zona em que a força de atrito estático supera o binário gerado pelo motor impedindo-o de rodar, assim como a zona de saturação do mesmo motor.

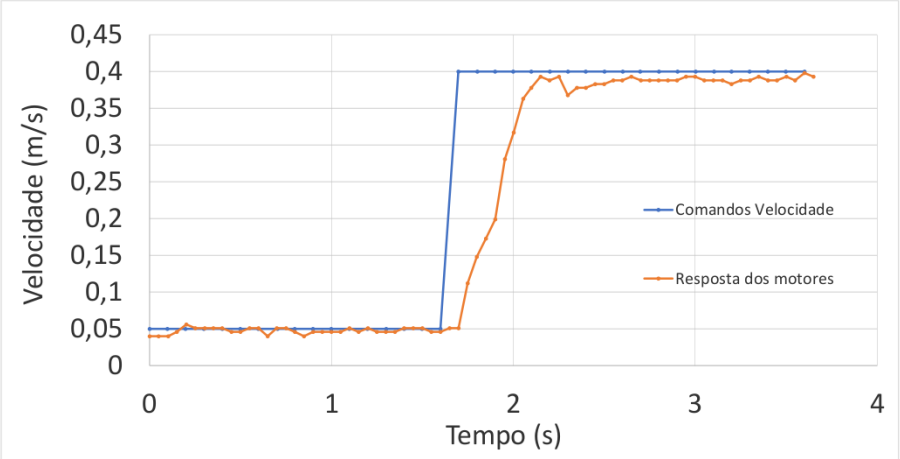


Figura 4.3: Resposta do motor a comandos de velocidade linear.

Para se obter a resposta da velocidade angular, escolheram-se os valores $0.34 rad/s$ e $1.36 rad/s$ (Figura 4.4).

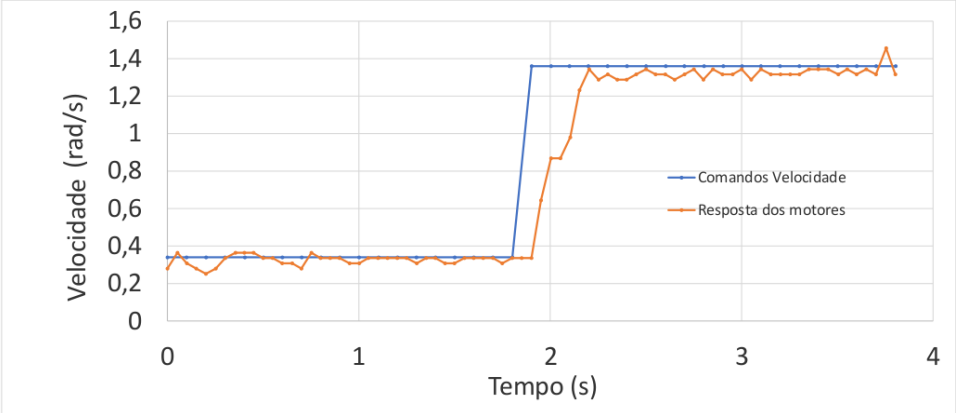


Figura 4.4: Resposta do motor a comandos de velocidade angular.

Como é possível observar nas figuras anteriores, a resposta do robô a comandos de velocidade é muito rápida, ou seja existem poucos pontos na região transitória. Deste modo, a parte derivativa dos controladores terá pouco ou nenhum efeito, sendo portanto descartada.

Nos controladores seguidores de trajetórias, a parte integral só é necessária se esta for uma curva. Como o percurso a efetuar pelo robô móvel é um segmento de reta, esta componente não terá efeito, sendo por isso nula. Como se pode observar nas figuras, o erro em regime permanente a uma referência de velocidade constante é praticamente nulo.

Desta forma, o sistema de controlo final terá só ganho proporcional.

Para a determinação das constantes do controlador será utilizado um método heurístico inspirado no método de malha fechada de *Ziegler–Nichols*. O ganho do controlador será incrementado até se atingir a instabilidade do sistema. Quando este se tornar instável, o ganho utilizado será metade desse valor [37, 38].

Iniciou-se o processo de calibração pela determinação do ganho proporcional para o controlador do erro de ângulo ($K_{p\Delta\theta}$). Para este conjunto de experiências, o robô foi inicializado sobre o segmento de reta, mas desalinhado como se observa na figura 4.5 e sem controlador para corrigir o erro de distância.

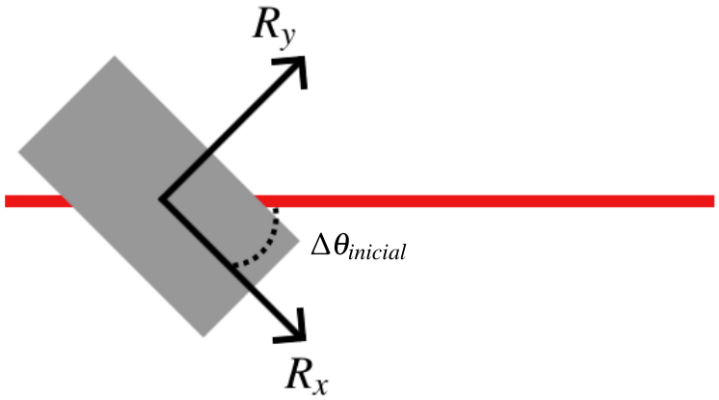


Figura 4.5: Exemplo da experiência efetuada para a determinação do ganho $Kp_{\Delta\theta}$ do controlador. $\Delta\theta_{inicial}$ representa o erro de ângulo inicial.

Começou-se por aumentar o ganho $K_{p\Delta\theta}$, tendo-se atingido o limite de instabilidade com K_p = 6.5 e obtendo-se uma frequência de oscilação entre os 1 e os 1.25 Hz, como é possível observar na figura 4.6.

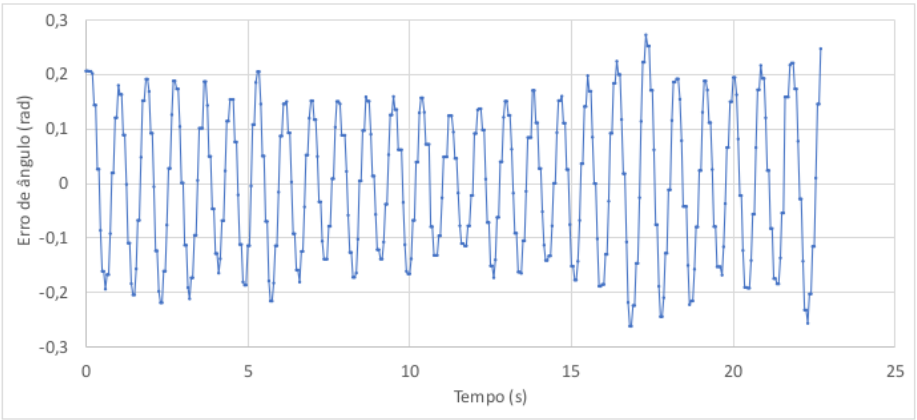


Figura 4.6: Evolução do erro de ângulo ao longo do tempo com ganho 6.5

Desta forma, realizou-se uma experiência com metade desse ganho.

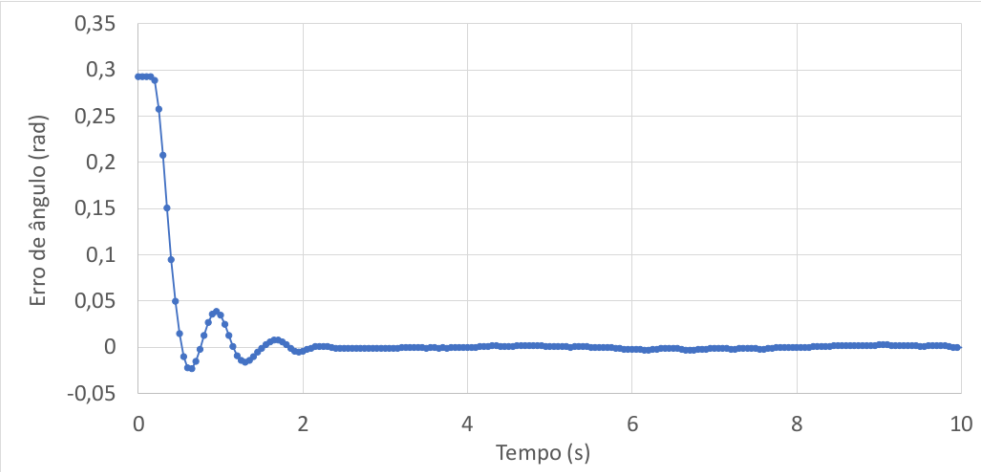


Figura 4.7: Evolução do erro de ângulo ao longo do tempo com ganho 3.25

Como é possível observar na figura 4.7, obteve-se uma sobre-elongação na resposta, desta forma, diminuiu-se um pouco K_p, de modo a eliminar esse efeito.

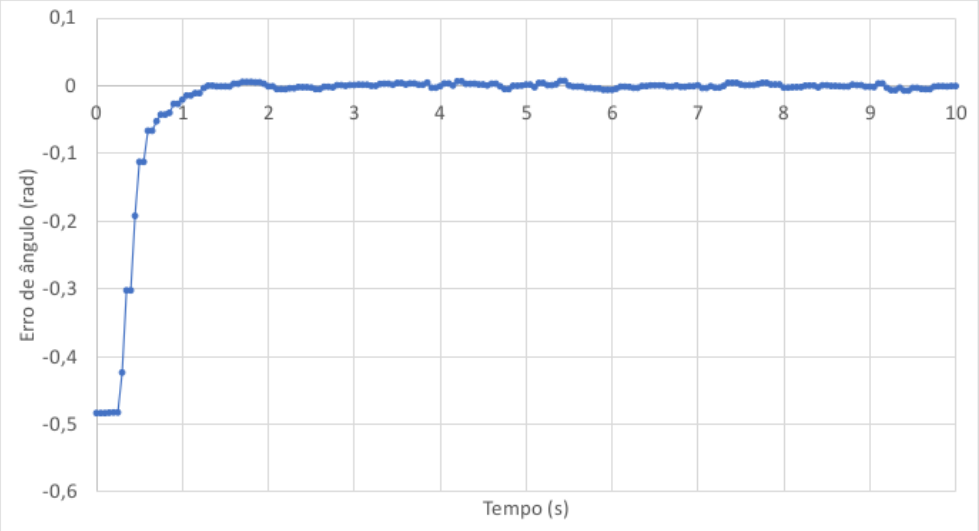


Figura 4.8: Evolução do erro de ângulo ao longo do tempo com ganho 2.5

Diminuindo o ganho até 2.5, foi possível obter uma resposta sem sobre-elongação, como é demonstrado na figura 4.8, com um tempo de estabelecimento para uma vizinhança de 2% do valor final da resposta de 1,25 segundos e para uma vizinhança de 5% do valor final da resposta de 1 segundo.

Para determinar o ganho para o controlador do erro de distância à reta ($Kp_{\Delta dist}$), manteve-se o ganho do controlador de ângulo determinado anteriormente. Para este conjunto de experiências, o robô foi inicializado com a mesma orientação da reta, mas afastado desta como é possível observar

na figura 4.9. A distância inicial à linha foi escolhida de forma a evitar que os motores atingissem a sua zona de saturação.

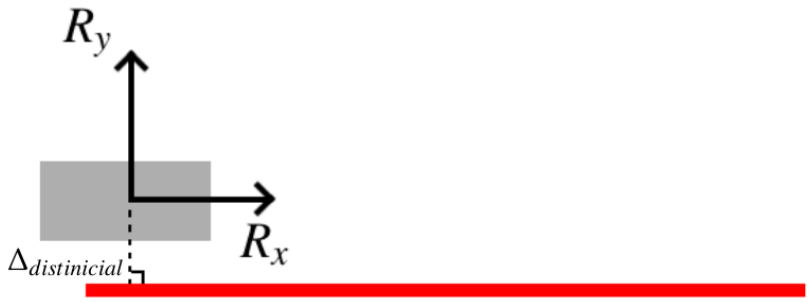


Figura 4.9: Exemplo da experiência efetuada para a determinação do ganho $K_{p\Delta dist}$ do controlador. $\Delta_{distinicial}$ representa o erro de distância inicial.

Para a determinação do ganho do controlador de distância, não se recorrerá à mesma abordagem utilizada para a calibração do do controlador de ângulo. De facto, a presença de velocidade linear no robô dificulta a aplicação do método de *Ziegler–Nichols*, assim como, na zona do limite de instabilidade, devido aos movimentos do robô móvel, a sua segurança pode não ser assegurada.

O ganho $K_{p\Delta dist}$ foi sendo aumentado até a resposta do sistema começar a apresentar sobre-elongação, obtendo-se um ganho de 17.5.

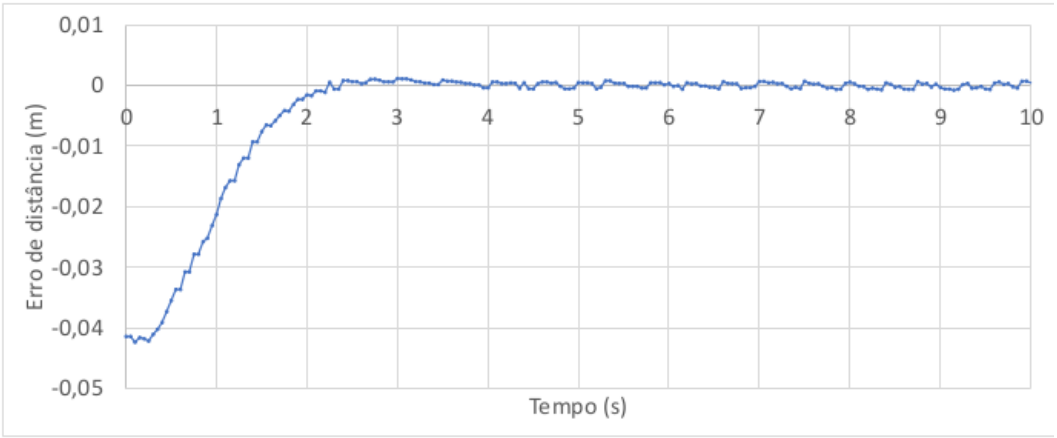


Figura 4.10: Evolução do erro de distância em função do tempo com ganho 17.5.

Observando a figura 4.10, o robô, para atingir o regime de estabilidade para uma vizinhança de 2% do valor final da resposta, demora 2.1 segundos, sendo que para uma vizinhança de 5% do

valor final da resposta demora 2 segundos e apresenta um erro absoluto final com uma amplitude máxima de cerca de 1 milímetro.

Capítulo 5

Algoritmos de deteção de marcos/balizas

Neste capítulo serão apresentados os algoritmos utilizados para a deteção da *docking station*.

5.1 Detetor de Linha

Durante esta dissertação, foi desenvolvido um método capaz de detetar uma linha através dos dados de um laser *rangefinder*. Este terá como objetivo aproximar por uma reta os pontos detetados por um laser que se encontram numa região de interesse. Mediante a informação recolhida, é determinado o ponto médio do segmento de reta, assim como a sua inclinação. Com estes dados, o algoritmo atribui um referencial a este marco (Figura 5.1). Para futuras referências, este método será denominado por "Detetor de Linha".

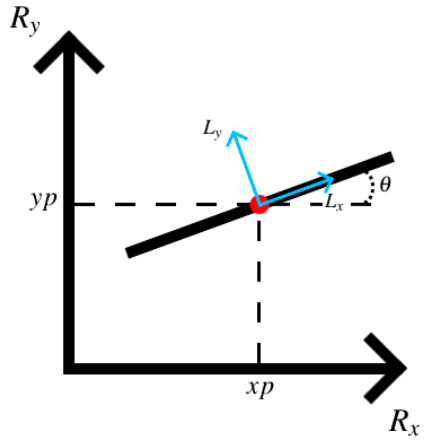


Figura 5.1: Exemplo de um resultado esperado. O ponto vermelho representa o ponto médio da linha com coordenadas (xp, yp) e θ a inclinação da mesma, ambos em relação ao referencial relativo (R_x, R_y), enquanto que (L_x, L_y) representam o referencial atribuído a este marco.

Um segmento de reta tem várias representações, sendo uma delas dada pela equação (5.1):

$$y = m \cdot x + b \tag{5.1}$$

onde m representa o declive desta e b o ponto de intersecção com o eixo das ordenadas. Apesar de nas retas horizontais o declive ser igual a 0, para retas verticais esta constante tem valor ∞, ou seja é uma indeterminação. Desta forma, quando a linha é aproximadamente vertical, para se evitar esta indeterminação, é aplicada aos pontos um rotação de -$\frac{\pi}{2}$ em torno do eixo Z do referencial, passando a linha a ser horizontal.

Através dos pontos obtidos após a rotação, é possível determinar as constantes que parametrizam a reta horizontal associada, permitindo determinar o ponto médio da linha assim como a sua inclinação. Tal possibilita a atribuição de um referencial ao marco horizontal. A este é aplicada uma rotação inversa da mencionada, atribuindo-o ao segmento de reta original.

De modo a determinar m e b foi utilizada uma abordagem recorrendo ao método dos mínimos quadrados.

$$Y_i = m \cdot X_i + b \ , \ i = 1,...,N \tag{5.2}$$

,onde N representa o número de pontos pertencentes à linha.

$$\begin{bmatrix} Y_1 \\ Y_2 \\ ... \\ Y_N \end{bmatrix} = \begin{bmatrix} X_1 & 1 \\ X_2 & 1 \\ ... & ... \\ X_N & 1 \end{bmatrix} \cdot \begin{bmatrix} m \\ b \end{bmatrix} \Leftrightarrow Y = X \cdot \Theta \tag{5.3}$$

A melhor estimativa, segundo o critério dos mínimos quadrados, é dada por:

$$\widehat{\Theta} = (X^T \cdot X)^{-1} \cdot X^T \cdot Y \tag{5.4}$$

Calculando $(X^T \cdot X)^{-1}$,

$$(X^T \cdot X)^{-1} = \left(\begin{bmatrix} X_1 & ... & X_N \\ 1 & ... & 1 \end{bmatrix} \cdot \begin{bmatrix} X_1 & 1 \\ ... & ... \\ X_N & 1 \end{bmatrix} \right)^{-1} \tag{5.5}$$

$$= \left(\begin{bmatrix} \sum_{i=1}^{N} X_i^2 & \sum_{i=1}^{N} X_i \\ \sum_{i=1}^{N} X_i & N \end{bmatrix} \right)^{-1} \tag{5.6}$$

$$= \frac{1}{N \cdot \sum_{i=1}^{N} X_i^2 - (\sum_{i=1}^{N} X_i)^2} \cdot \left(\begin{bmatrix} N & -\sum_{i=1}^{N} X_i \\ -\sum_{i=1}^{N} X_i & \sum_{i=1}^{N} X_i^2 \end{bmatrix} \right) \tag{5.7}$$

Na equação (5.7), é preciso verificar se $N \cdot \sum_{i=1}^{N} X_i^2 - (\sum_{i=1}^{N} X_i)^2$ é muito próximo de 0. Caso se verifique esta situação, não será possível determinar a reta, visto que ou não existem pontos suficientes ou que estes se encontram muito próximos.

Calculando $X^T \cdot Y$,

$$X^T \cdot Y = \begin{bmatrix} X_1 & \dots & X_N \\ 1 & \dots & 1 \end{bmatrix} \cdot \begin{bmatrix} Y_1 \\ \dots \\ Y_N \end{bmatrix} \tag{5.8}$$

$$= \begin{bmatrix} \sum_{i=1}^{N} X_i \cdot Y_i \\ \sum_{i=1}^{N} Y_i \end{bmatrix} \tag{5.9}$$

Substituindo as equações 5.7 e 5.9 na equação 5.4.

$$\widehat{\Theta} = \frac{1}{N \cdot \sum_{i=1}^{N} X_i^2 - (\sum_{i=1}^{N} X_i)^2} \cdot \left(\begin{bmatrix} N & -\sum_{i=1}^{N} X_i \\ -\sum_{i=1}^{N} X_i & \sum_{i=1}^{N} X_i^2 \end{bmatrix} \right) \cdot \begin{bmatrix} \sum_{i=1}^{N} X_i \cdot Y_i \\ \sum_{i=1}^{N} Y_i \end{bmatrix} \tag{5.10}$$

Aplicou-se uma rotação de θ em torno do eixo Z aos pontos do laser (Figura 5.2), assim como ao ponto inicial da reta (para $x = 0$), determinado através das constantes m e b. Deste modo, os pontos ficam alinhados com o eixo das abcissas.

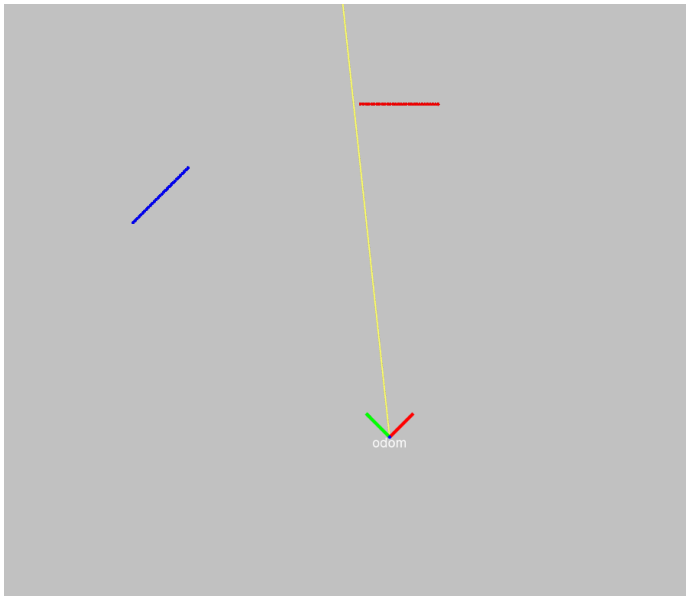


Figura 5.2: Rotação dos pontos do laser em relação ao eixo Z do referencial . No referencial, o eixo vermelho representa o eixo das abcissas e o verde o das ordenadas. Os pontos vermelhos representam os dados medidos pelo laser e os azuis os resultantes de uma rotação de θ em torno do eixo Z do referencial. (Imagem tirada do visualizador 3D Rviz).

Fazendo uma média das abcissas de todos os pontos do sensor é possível determinar a coordenada x do ponto médio. A ordenada do ponto médio será igual à coordenada y do ponto já calculado (para $x = 0$) após a rotação. Obtidas as suas coordenadas aplica-se uma rotação inversa, de forma a obter o ponto médio em relação ao referencial original (Figura 5.3).

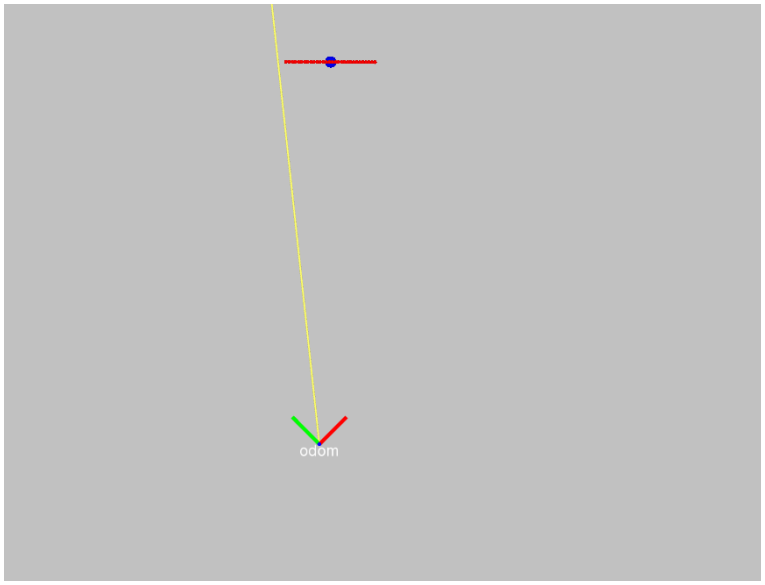


Figura 5.3: Ponto médio da Linha. No referencial, o eixo vermelho representa o eixo das abcissas e o verde o das ordenadas. Os pontos vermelhos representam os dados medidos pelo laser e o azul o ponto médio da linha. (Imagem tirada do visualizador 3D Rviz)

O algoritmo 1 descreve o detetor implementado. O detetor tem como entrada os dados obtidos pelo laser ($Z_L(k)$), uma posição estimada onde se encontra a reta ($P(k)$) e o tamanho da linha que se pretende detetar (*tamanhoLinha*). Já como saída, tem a posição do referencial atribuído à linha ($\chi(k)$) bem como a sua orientação expressa num quaternião ($q(k)$).

Algorithm 1: Detetor de Linha

Input: $Z_L(k)$, $P(k)$, $tamanhoLinha$

Output: $\chi(k)$, $q(k)$

begin

- LinhaVertical = false;
- $Z_{Lp}(k)$ = ClusterizarData($Z_L(k)$, $P(k)$, $tamanhoLinha$);
- **if** *VerificarSeVertical*($Z_{Lp}(k)$) **then**
  - LinhaVertical = true;
  - $Z_{Lp}(k)$ = RodarPontos($Z_{Lp}(k)$, $\frac{-\pi}{2}$);
- **end**
- $\begin{bmatrix} m \\ b \end{bmatrix}$ = DeterminarConstantesLinha($Z_{Lp}(k)$);
- y_{aux} = DeterminarPonto(m,b,0);
- $Z_{Lp}(k)$ = RodarPontos($Z_{Lp}(k)$,$atan2(-m,1)$);
- $\begin{bmatrix} x_{aux} \\ y_{aux} \end{bmatrix}$ = RodarPontos($\begin{bmatrix} 0 \\ y_{aux} \end{bmatrix}$,$atan2(-m,1)$);
- $\begin{bmatrix} X_p \\ Y_p \end{bmatrix}$ = DeterminarPontoMédio($Z_{Lp}(k)$, $\begin{bmatrix} x_{aux} \\ y_{aux} \end{bmatrix}$);
- $\begin{bmatrix} X_p \\ Y_p \end{bmatrix}$ = RodarPontos($\begin{bmatrix} X_p \\ Y_p \end{bmatrix}$,$atan2(m,1)$);
- $\theta_p = atan2(m,1)$;
- $\begin{bmatrix} \chi(k) \\ q(k) \end{bmatrix}$ = AtribuirReferencial($\begin{bmatrix} X_p \\ Y_p \end{bmatrix}$,$\theta_p$);
- **if** *LinhaVertical* **then**
  - $\begin{bmatrix} \chi(k) \\ q(k) \end{bmatrix}$ = RodarReferencial($\begin{bmatrix} \chi(k) \\ q(k) \end{bmatrix}$,$\frac{\pi}{2}$);
- **end**

end

A linha 3 do algoritmo 1 cria um cluster ($Z_{Lp}(k)$) com os pontos do laser que se encontrem dentro de uma zona de interesse, definida pelo ponto inicial ($P(k)$) (Figura 5.4). Esta zona tem o formato de um quadrado de lado igual a duas vezes o tamanho da linha que se pretende detetar (*tamanhoLinha*) e centrado no ponto inicial ($P(k)$). Foi escolhida esta configuração de modo a não restringir demasiado a colocação da *docking station*.

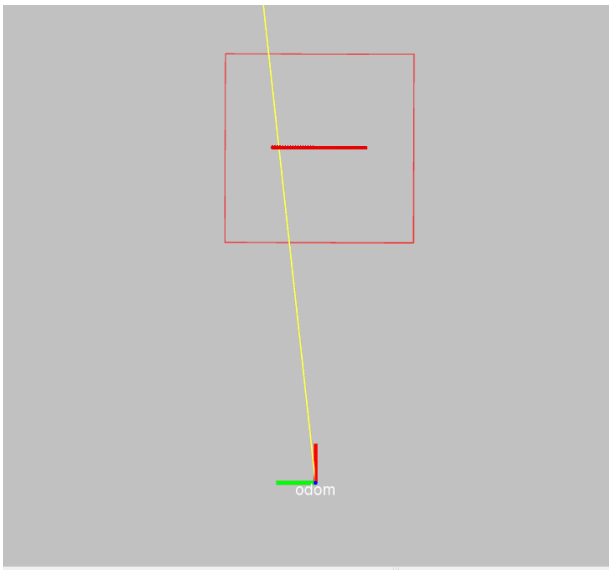


Figura 5.4: Zona de Interesse. No referencial, o eixo vermelho representa o eixo das abcissas e o verde o das ordenadas. Os pontos vermelhos representam os dados medidos pelo laser. O quadrado a vermelho representa a zona de interesse, sendo que qualquer ponto fora desta zona será ignorado (Imagem tirada do visualizador 3D Rviz).

Na linha 4 é determinado se a linha é vertical. Para tal, calcula-se a média da coordenada x dos pontos do cluster e verifica-se se 90% dos pontos se encontram dentro de um *threshold* em torno do valor calculado. Sendo vertical, na linha 6 é aplicada uma rotação de $\frac{-\pi}{2}$ aos pontos do laser, para assim ser possível determinar as constantes que parametrizam a reta.

Na linha 8 são determinadas as constantes através da equação 5.10.

Na linha 9 é calculado o ponto para $x = 0$, que será depois utilizado para determinar a ordenada do ponto médio.

Nas linhas 10 e 11, aos pontos do laser e ao ponto determinado, é aplicada uma rotação de forma a ficarem alinhados com o eixo das abcissas. Na linha 12 são determinadas as coordenadas do ponto médio da reta, sendo que na linha 13 é aplicada uma rotação inversa a esse ponto.

Na linha 15 é atribuído um referencial ao marco, utilizando o ponto médio determinado assim como a inclinação da reta.

Na linha 16 é verificado se os pontos iniciais formavam uma linha vertical. Se isto se verifica, é aplicada uma rotação ao referencial, de forma a este ser atribuído à linha original (Figura 5.5).

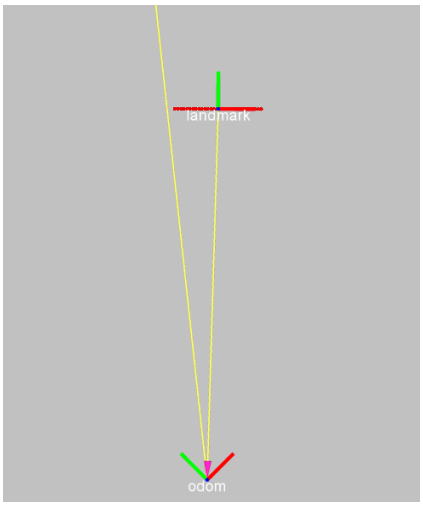


Figura 5.5: Referencial no objeto. (Imagem tirada do visualizador 3D Rviz)

5.2 Localização de Beacons

H. Sobreira *et al.* [39, 40] apresentam um algoritmo que consiste na estimação da pose de um veículo autónomo $\chi_R = [x_R, y_R, \theta_R]^T$ no referencial global $[W_x, W_y]$.

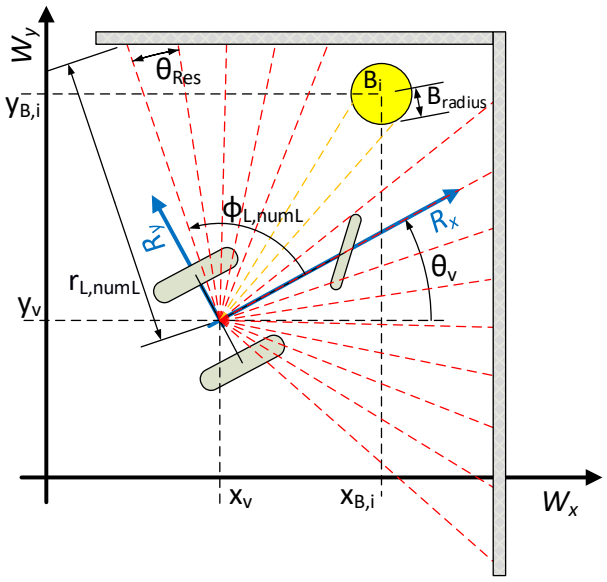


Figura 5.6: Representação do robô móvel no referencial global (W_x, W_y). O círculo amarelo representa um refletor numa posição fixa $[x_{B,i}, y_{B,i}]$. As linhas tracejadas representam medidas obtidas pelo laser, sendo que as linhas a vermelho representam medidas com baixa refletividade e as linhas a laranja com alta refletividade (Adaptado de [40]).

Numa varredura do laser de segurança é retornado um conjunto de pontos em coordenadas polares $[\rho_i, \theta_i]$ dos objetos detetados em relação ao referencial relativo do robô móvel $[R_x, R_y]$. A cada medida está associada uma variável booleana relacionada com o índice de reflexão do corpo.

De facto, se este índice for alto a variável tem valor 1, caso contrário tem valor 0. Como os *beacons* apresentam um índice de reflexão elevado, estes serão facilmente distinguidos do ambiente em redor. Desta forma, conhecendo a posição dos *beacons* em relação ao referencial Global será possível determinar a pose do robô móvel.

O algoritmo implementado tem como entradas os dados obtidos pelo laser, um mapa com a posição dos *beacons* e os valores de odometria obtidos através dos encoders. Este é dividido em 4 módulos, como é possível observar na figura 5.7.

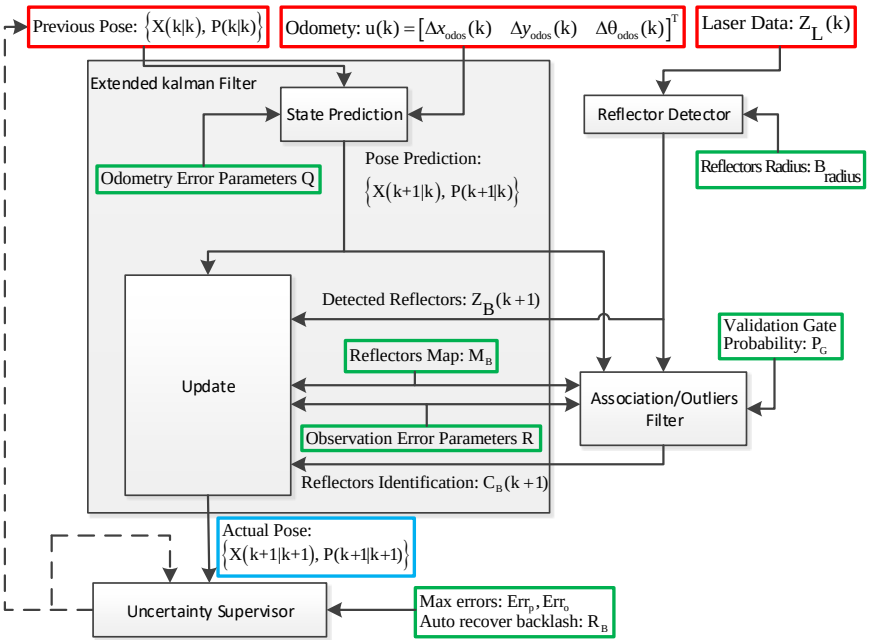


Figura 5.7: Diferentes módulos que compõem a arquitetura do sistema (blocos pretos). Os blocos vermelhos representam as entradas do sistema, a verde os parâmetros e a azul a saída (Adaptado de [40]).

O sistema baseia-se na aplicação de um algoritmo de fusão sensorial com a odometria. Para tal, é utilizado um filtro de Kalman Extendido (EKF), que funde a informação processada do laser (distância e ângulo dos marcadores detetados) com a odometria.

O módulo do detetor de *beacons* determina a distância e ângulo de possíveis marcadores em relação ao referencial relativo. No entanto, os *beacons* não são os únicos objetos com índice de reflexão elevados. Desta forma, os autores implementaram um filtro para eliminar estes *outliers*. Para tal, associam o número de pontos que refletem no corpo com o formato do *beacon* pretendido. Sendo este cilíndrico, com um raio conhecido, é possível filtrar certas reflexões indesejáveis.

O módulo *Association \ outliers filter* identifica os beacons processados pelo módulo anterior, através da posição estimada do robô e da sua covariância determinadas pelo EKF. Durante esse processo, elimina *outliers* que não tenham sido filtrados anteriormente.

O último módulo serve para supervisionar o sistema, para caso a matriz de covariância atinja valores elevados, o robô móvel pare até que esta diminua. Isto acontece quando a estimação da pose do veículo não é fiável.

Mais detalhes sobre o algoritmo podem ser encontrados em [39].

5.2.1 Modificações feitas

Como já foi referido, o módulo refletor de *beacons* foi desenvolvido de forma a detetar *beacons* cilíndricos. Como tal, foi desenvolvido um módulo capaz de detetar *beacons* planares. Este terá como entradas os dados do Laser ($Z_L(k)$) e o tamanho dos beacons a detetar (*tamanhoBeacon*).

O algoritmo 2 descreve o módulo:

Algorithm 2: Detetor de *Beacons* planos

Input: $Z_L(k)$, *tamanhoBeacon*
Output: $Z_B(k)$

```
1 begin
2     Z_Lc(k) = ClusterizarData(Z_L(k));
3     Z_C(k) = FiltrarClusters(Z_Lc(k), tamanhoBeacon);
4     for todosClusters(Z_C(k)) do
5         [x_p; y_p] = DeterminarPontoMedioLinha(Z_C(k));
6         adicionar [x_p; y_p] em Z_B(k)
7     end
8 end
```

A linha 2 do algoritmo 2 divide os dados obtidos pelo laser em *clusters*, tendo em conta o nível de intensidade do índice de reflexão. Um *cluster* Z_{Lc} é constituído por pontos adjacentes com um índice de reflexão elevado.

Na linha 3, esses *clusters* são filtrados. Por outras palavras, é avaliada a dimensão de cada um e caso esta esteja dentro da gama $tamanhoBeacon \pm 30\%$, o conjunto de pontos é inserido em $Z_C(k)$. Se o *cluster* tiver um tamanho superior a $tamanhoBeacon + 30\%$, este é descartado, no entanto, se apresentar um tamanho inferior a $tamanhoBeacon - 30\%$, este é guardado numa variável auxiliar. Caso o *cluster* seguinte também apresente um tamanho reduzido, a distância entre os dois *clusters* será calculada. Se eles forem suficientemente próximos então são fundidos. Seguiu-se esta abordagem para o caso de o laser obter uma medida defeituosa num refletor, onde o índice de reflexão deveria ser alto, porém apresentou um valor baixo. Tal leva à criação de dois *clusters* de tamanho reduzido.

Na linha 5, cada *cluster* identificado é percorrido, sendo que o seu ponto médio é determinado e na linha 6 o ponto é inserido em $Z_B(k)$.

5.3 Perfect Match

H. Sobreira *et al.*[40, 41] efetuaram modificações ao algoritmo *Perfect Match* de Martin Lauer et al. [11], de forma a ser possível utilizá-lo em robôs móveis equipados com lasers.

O sistema é composto por dois grandes módulos: um onde se encontram os componentes do filtro de Kalman e um outro onde estão presentes os constituintes do algoritmo de ajuste de mapa. Este último tem como objetivo minimizar uma função de custo, que lhe permita obter a melhor correspondência entre um template e a informação obtida através do sensor laser, utilizando um método numérico denominado por *Resilient BackPropagation* (RPROP).

A cada dado do laser é calculada a distância entre este e a célula ocupada mais próxima do mapa (D_i). Este desvio é penalizado pela função do erro (equação 5.11).

$$err_i(D_i) = 1 - \frac{L_c^2}{L_c^2 + D_i^2} \tag{5.11}$$

,onde L_c é um parâmetro que permite limitar o peso de *outliers*. Devido à posição do laser de segurança (perto do solo), a existência de elementos não mapeados como pessoas, cadeiras e mobília, entre outros é muito elevada. Os autores, para diminuir o efeito destas medidas erróneas, limitam o erro de distância (MaxErrorDist), sendo que se ultrapassar um limite definido, as medidas são ignoradas. Para além desta alteração a função erro é multiplicada por um fator de distância, dando mais peso a medidas mais distantes.

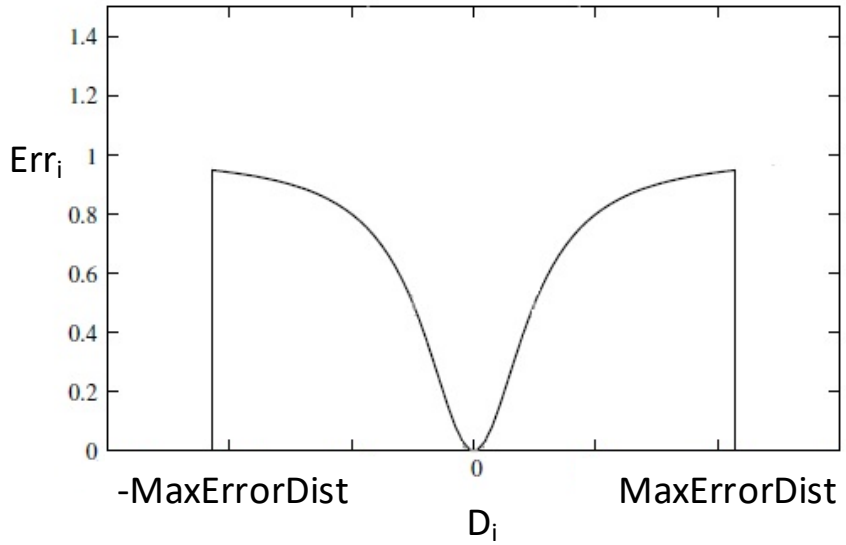
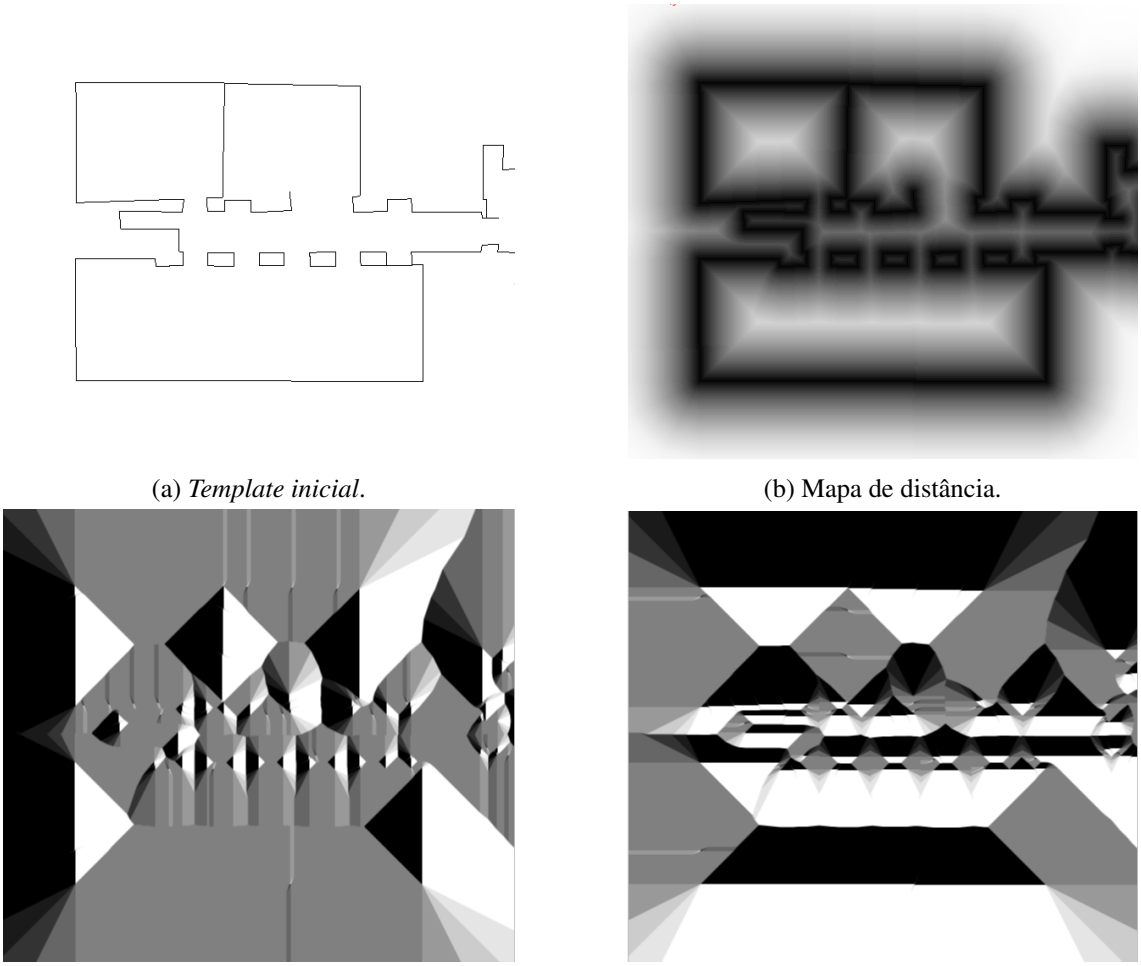


Figura 5.8: Função de custo limitada (Adaptada de [40]).

O algoritmo RPROP baseia-se no sinal das primeiras derivadas da função de custo, tendo sido desenvolvido para problemas de otimização não lineares. Este calcula iterativamente a pose do robô que melhor minimiza a função custo.

De modo a minimizar o tempo de processamento, as derivadas são calculadas à priori e colocadas numa *lookup table*. Para tal, são calculadas a matriz de distância e, a partir desta, as matrizes

de gradiente ao longo do eixo x e do eixo y.



(a) *Template inicial*.

(b) Mapa de distância.

(c) Gradiente ao longo do eixo de X.

(d) Gradiente ao longo do eixo de Y.

Figura 5.9: Exemplo de *lookup tables* (Adaptadas de [40]).

Para esta aplicação, foi utilizada a fase de predição de estado do filtro de *Kalman* e o algoritmo de ajuste de mapa mencionado.

5.3.1 Modificações efectuadas

Dado que o sensor laser é um sensor radial, implica que o número de pontos que atinge o marco da *docking station* reaja de forma inversamente proporcional com a distância deste ao robô móvel. De facto, quanto menor for a distância, maior é o número de pontos incidentes no marco. Desta forma, a resposta do *Perfect Match* estabiliza, à medida que o robô móvel se aproxima da *docking station*, como é possível observar na figura 5.10.

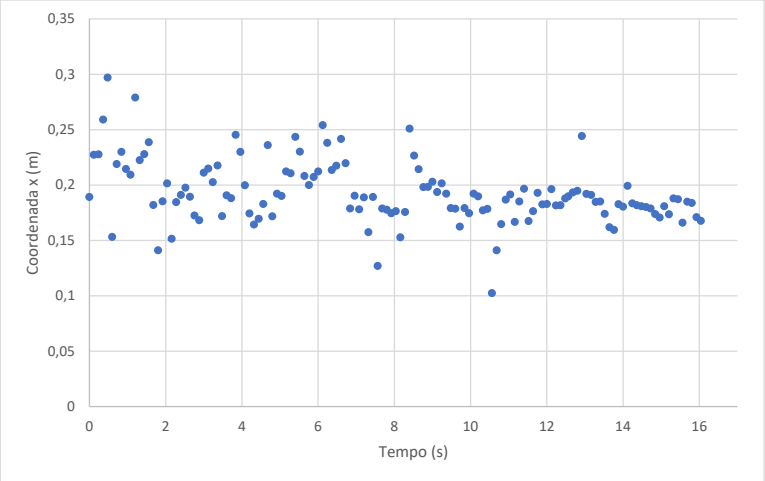


Figura 5.10: Evolução da coordenada x do referencial do marco onde se pretende docar. O robô móvel efetua uma aproximação ao objeto em questão.

Adicionalmente, de modo a diminuir a variação que existe, implementou-se um filtro passa-baixo à resposta do algoritmo (equação 5.12).

$$y[n] = y[n-1] \cdot \alpha + x[n] \cdot (1-\alpha), \alpha \in [0,1] \tag{5.12}$$

onde y[n] representa a saída atual, y[n-1] a saída no instante anterior, x[n] o resultado atual obtido a partir do *Perfect Match* e α o peso dado às medidas anteriores.

Deslocando-se o robô móvel a uma velocidade linear v, este demorará Δt segundos a percorrer uma distância d. Desta forma, admitindo que se pretende que a resposta estabilize em metade do tempo, e que τ é aproximadamente $\frac{1}{4}$ do tempo de estabelecimento, obtém-se que:

$$\tau = \frac{1}{4} \cdot \frac{\Delta_t}{2} \tag{5.13}$$

A função de transferência deste filtro será

$$\frac{Y(s)}{U(s)} = \frac{1}{\tau \cdot s + 1} \tag{5.14}$$

e com pólo (σ)

$$\sigma = -\frac{1}{\tau} \tag{5.15}$$

Discretizando o pólo.

$$Z_p = e^{\sigma \cdot T_a} \Leftrightarrow Z_p = e^{-\frac{T_a}{\tau}} \tag{5.16}$$

, em que T_a corresponde ao tempo de amostragem, que neste caso será de 0.1 segundo, que corresponde à taxa de amostragem do laser de segurança. Desta forma, o valor utilizado na equação do filtro (α) será igual ao pólo em tempo discreto (Z_p).

Capítulo 6

Análise de Resultados

Neste capítulo serão analisados os resultados obtidos utilizando os robôs *Jarvis* e *Edgar*.

6.1 Resultados obtidos com o Jarvis

Numa operação de atracagem é fundamental assegurar que o robô móvel se encaminha com precisão e repetibilidade para o objeto no qual pretende atracar. Durante esta operação, o robô deverá detetar o alvo e encaminhar-se para o mesmo. De forma a avaliar a robustez da deteção do objeto, foram realizadas várias experiências com o *Jarvis* (secção 3.5.2), testando-se o algoritmo de localização de *Beacons* (Secção 5.2), o algoritmo de deteção de linha (Secção 5.1) e o algoritmo *Perfect Match* (Secção 5.3), sendo avaliada a repetibilidade na pose final do veículo.

Procedeu-se à experiência do seguinte modo: o robô móvel foi inicializado em pontos aleatórios dentro de uma área onde este conseguisse ver a face da *dock* onde iria atracar e se encontrasse afastado desta (aproximadamente 5 m). Como é possível observar na figura 6.1.

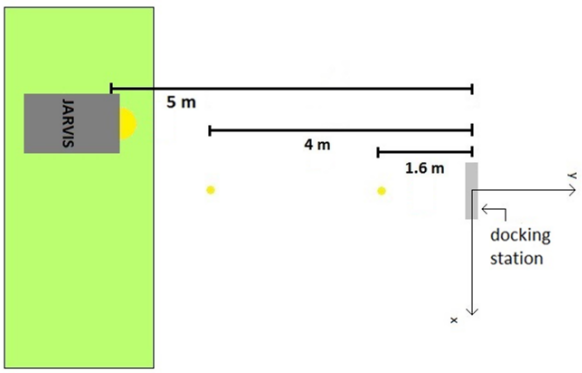


Figura 6.1: Exemplo da experiência descrita. A área a verde corresponde a posições possíveis de onde o robô pode partir.

Nesta fase de testes, o robô dirige-se para uma posição afastada do objeto (cerca de 4 metros), mas alinhada com esta, orientando-se de seguida para este. Posteriormente, descreverá uma

trajetória linear com o objetivo de se aproximar da *docking station*. Quando esta é detetada, o algoritmo utilizado atribui-lhe um referencial. Esse será usado para alinhar o robô segundo o eixo Y, sendo que para esta fase de testes ficará a 1.6 metros da origem.

O *Jarvis* possui dois lasers, um de segurança e um de navegação, sendo que a pose final foi extraída com recurso a este último. Após 10 experiências, determinou-se a média das poses recorrendo à equação (6.1).

$$\mu_v = \frac{1}{N}\sum_{i=1}^{N} v_i \tag{6.1}$$

onde v representa as coordenadas do robô (x,y) e a sua orientação (θ), N representa o número de experiências efetuadas (neste caso foram efetuadas 10) e μ_v a média das variáveis referidas.

De forma a ser possível comparar os 3 métodos utilizou-se a equação (6.2), que representa a média dos erros absolutos obtidos, e a equação (6.3), que permite determinar o desvio padrão:

$$\mu_{erro} = \frac{1}{N}\sum_{i=1}^{N} |\mu_v - v_i| \tag{6.2}$$

$$\sigma_v = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (\mu_v - v_i)^2} \tag{6.3}$$

onde v representa as coordenadas do robô (x,y) e a sua orientação (θ), N representa o número de experiências efetuadas (neste caso foram efetuadas 10) e μ_v a média das variáveis referidas.

6.1.1 Precisão de execução da trajetória em linha reta

Numa primeira fase, decidiu-se testar o controlador implementado no *Jarvis*. Esta fase de testes serviu para verificar qual a precisão do controlador bem como a precisão do sistema de localização deste robô.

Para tal, o robô móvel percorreu uma trajetória linear, seguindo uma reta definida no referencial global, sendo que as suas poses finais foram utilizadas para avaliar o controlador. O veículo autónomo foi inicializado sempre no princípio da linha em posições laterais à mesma, como é possível observar na figura 6.2.

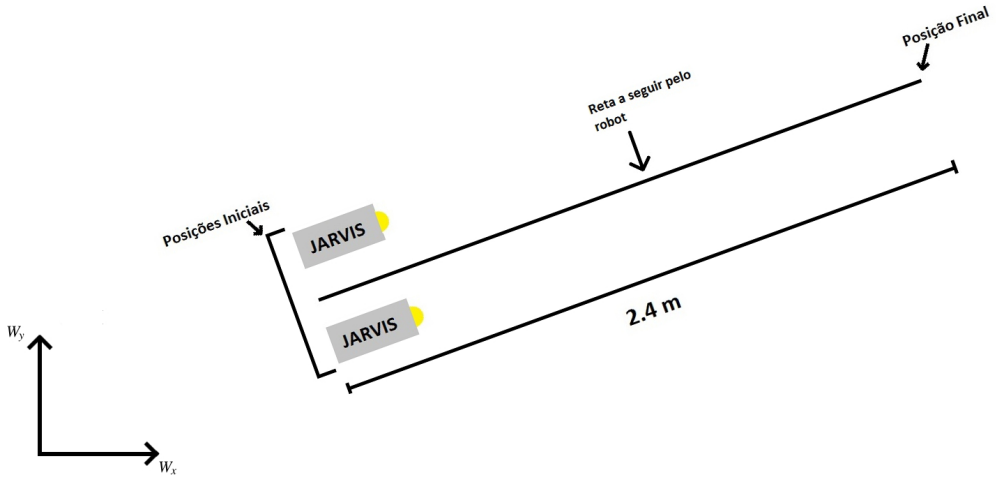
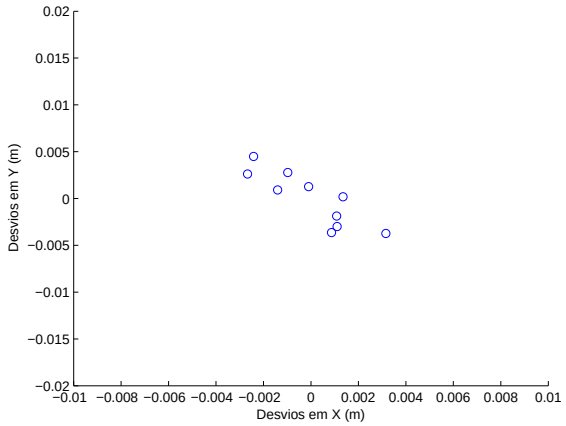


Figura 6.2: Possíveis posições iniciais do AGV.

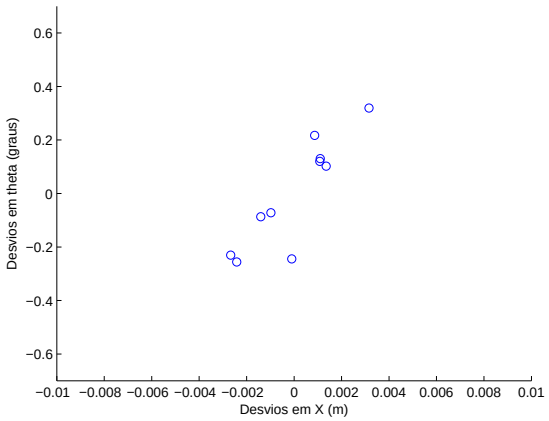
Após um conjunto de 10 experiências, foram obtidos os resultados presentes na tabela 6.1 e na figura 6.3.

Tabela 6.1: Média dos erros absolutos e desvio padrão das poses finais seguindo uma linha fixa

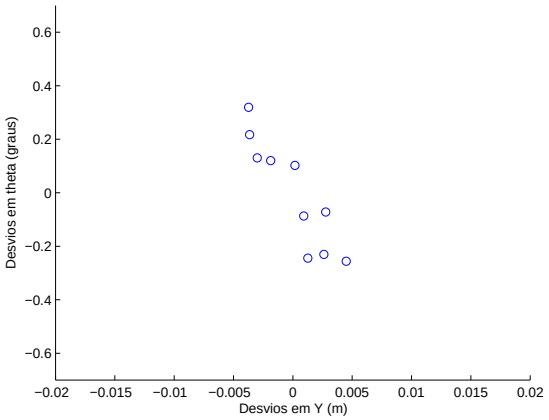
	μ_{erro}	σ
x	0,0015	0,0018
y	0,0024	0,0028
θ**(rad)**	0,0031	0,0034
$\theta(^{\mathbf{o}})$	0,178	0,195



(a) Variação dos desvios de Y (m) em relação aos desvios de X (m)



(b) Variação dos desvios de theta (graus) em relação aos desvios de X (m)



(c) Variação dos desvios de theta (graus) em relação aos desvios de Y (m)

Figura 6.3: Desvios observados em x e y e orientação testando o controlador do *Jarvis*

Observando a figura 6.3a, é possível constatar que existe uma correlação entre o desvio no eixo X e o desvio em Y, sendo estes inversamente proporcionais entre si. Isto acontece devido à forma como os erros do controlador seguidor de linha se propagam. De facto, estes apresentam uma maior dispersão transversal do que longitudinal. Como as poses finais estão a ser observadas em relação ao referencial Global, estes desvios serão referidos para esse referencial. Deste modo, é esperado que os desvios apresentem uma correlação inversamente proporcional entre si (Figura 6.4).

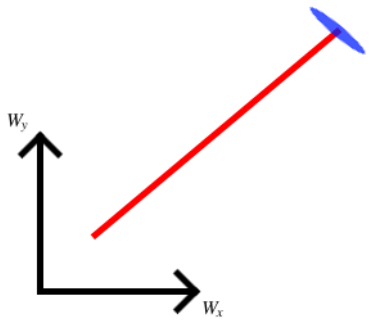


Figura 6.4: Forma como os desvios se dispersam em relação ao referencial Global (W_x, W_y). A vermelho representa a linha a percorrer e a elipse a azul a dispersão dos erros do controlador.

Como é possível observar na figura 6.3b, com o aumento do desvio em X, há um aumento do desvio em θ no mesmo sentido. Na última figura (6.3c), os desvios apresentam uma correlação inversamente proporcional.

6.1.2 *Docking station* com *beacons* refletores

No caso do algoritmo de localização de *Beacons*, foram colocadas duas faixas refletoras numa *dockstation* distanciadas de 38 cm entre si e com uma largura de 7 cm, como é vísivel na figura 6.5.

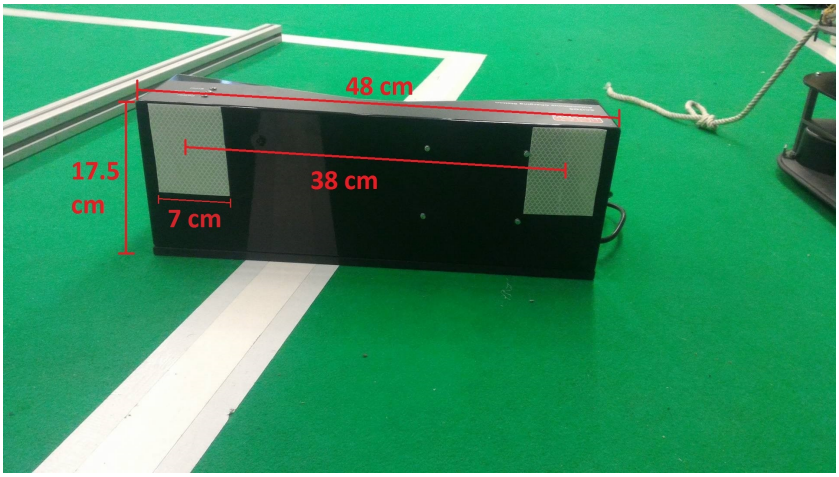


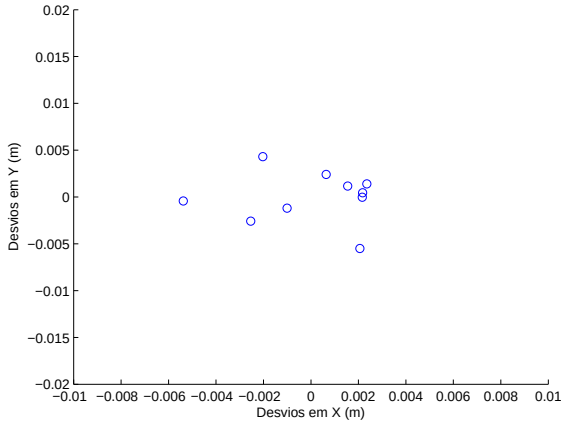
Figura 6.5: Dock com faixas refletoras para o Algoritmo de Localização de *Beacons*

Para este algoritmo, foram realizados dois conjuntos de experiências: um primeiro onde se utilizou o algoritmo com as modificações descritas na Secção 5.2.1 e outro sem as referidas modificações.

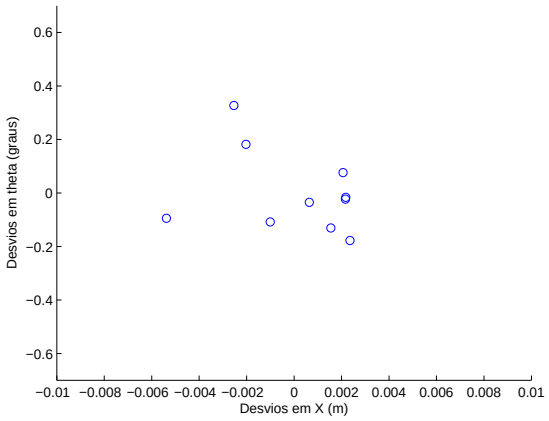
Após vários testes, foram obtidos os resultados presentes na tabela 6.2 e nas figuras 6.6 e 6.7 .

Tabela 6.2: Média dos erros absolutos e desvio padrão das poses finais utilizando o algoritmo de localização de *Beacons* sem e com modificações

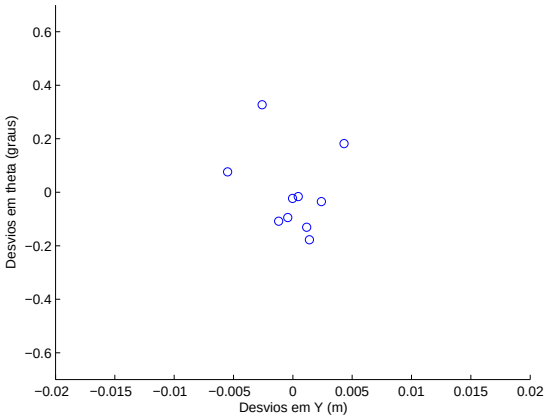
	μ_{erro}		σ	
	Sem Modificações	**Com Modificações**	**Sem Modificações**	**Com Modificações**
x	0,0022	0,0017	0,0025	0,0020
y	0,0019	0,0021	0,0026	0,0027
θ**(rad)**	0,0020	0,0027	0,0026	0,0036
$\theta(^{o})$	0,117	0,154	0,147	0,208



(a) Variação dos desvios de Y (m) em relação aos desvios de X (m)



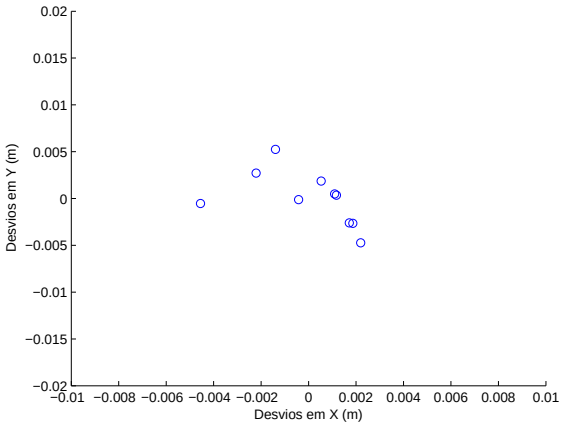
(b) Variação dos desvios de theta (graus) em relação aos desvios de X (m)



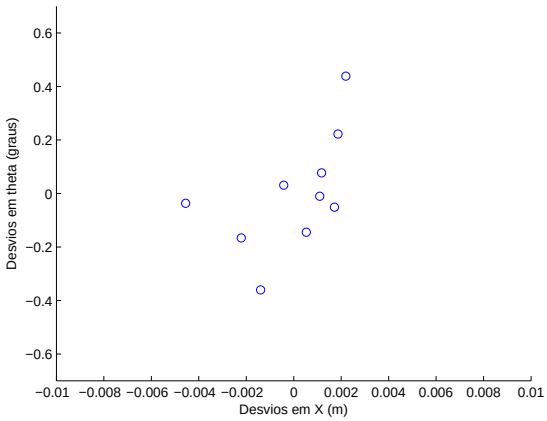
(c) Variação dos desvios de theta (graus) em relação aos desvios de Y (m)

Figura 6.6: Desvios observados em x e y e orientação utilizando o algoritmo de localização de *Beacons* sem modificações

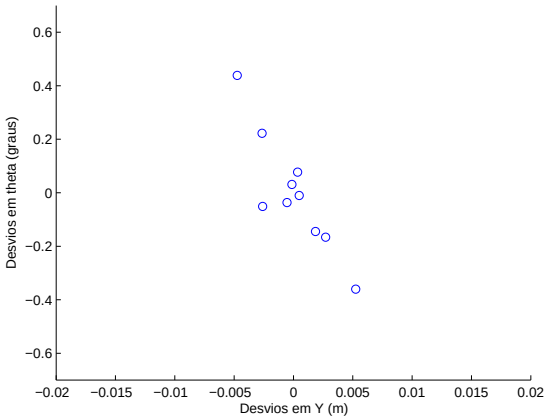
Observando a figura 6.6, é possível reparar que não existe qualquer correlação entre os diferentes desvios.



(a) Variação dos desvios de Y (m) em relação aos desvios de X (m)



(b) Variação dos desvios de theta (graus) em relação aos desvios de X (m)



(c) Variação dos desvios de theta (graus) em relação aos desvios de Y (m)

Figura 6.7: Desvios observados em x e y e orientação utilizando o algoritmo de localização de *Beacons* com modificações

Nos dois primeiros gráficos, é possível reparar que os diferentes desvios não se correlacionam, no entanto, na figura 6.7c, constata-se que estes apresentam uma correlação inversamente proporcional.

Em relação aos dados obtidos, não é possível tirar nenhuma conclusão em concreto, visto que os erros encontram-se na ordem de grandeza dos erros de posicionamento do controlador de trajetórias do robô. Ou seja conclui-se que o erro é dessa mesma ordem de grandeza.

6.1.3 Detetor de Linha

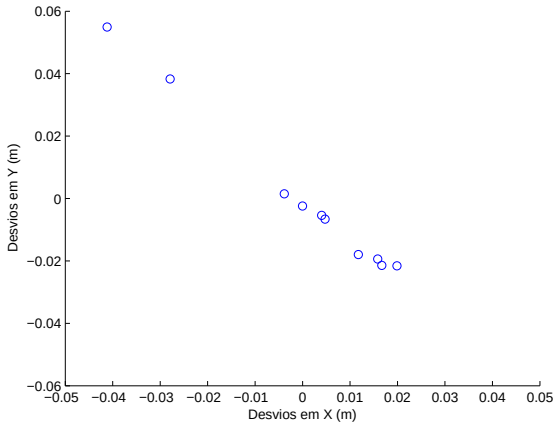
Nos testes com o algoritmo Detetor de Linha foram usados 2 objetos com comprimentos diferentes: uma chapa de 17.5 cm (Figura 6.8) e a *docking station* de 48 cm (Figura 6.5).

Figura 6.8: Chapa com 17.5 cm de comprimento

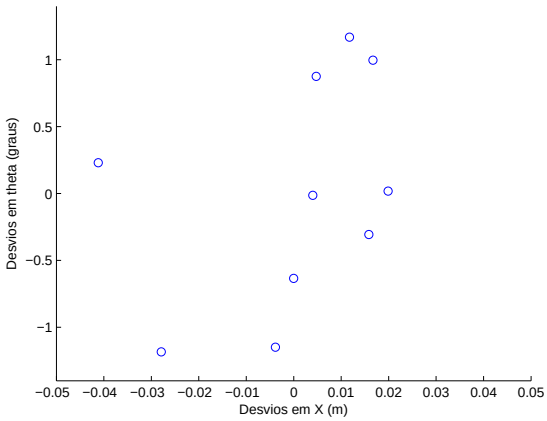
Após vários testes, foram obtidos os resultados presentes na tabela 6.3 e nas figuras 6.9 e 6.10.

Tabela 6.3: Média dos erros absolutos e desvio padrão das poses finais utilizando o algoritmo detetor de linhas

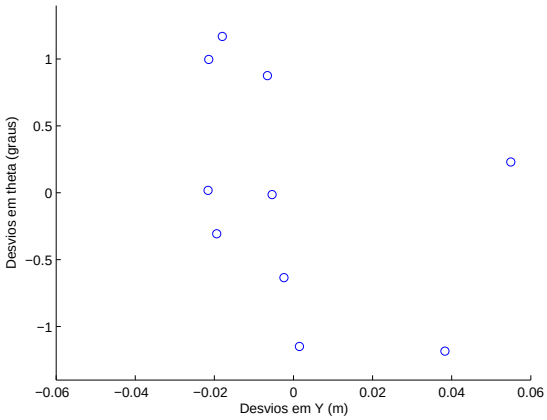
	μ_{erro}		σ	
	***Docking station* 48 cm**	**Chapa 17.5cm**	***Docking station* 48 cm**	**Chapa 17.5 cm**
x(m)	0.0042	0.0146	0.0047	0.0190
y(m)	0.0042	0.0190	0.0048	0.0249
θ**(rad)**	0.0088	0.0115	0.0117	0.0140
$\theta(^{o})$	0.501	0.658	0.673	0.800



(a) Variação dos desvios de Y (m) em relação aos desvios de X (m)



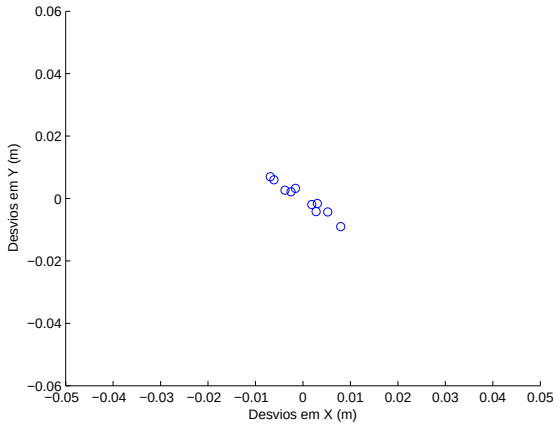
(b) Variação dos desvios de theta (graus) em relação aos desvios de X (m)



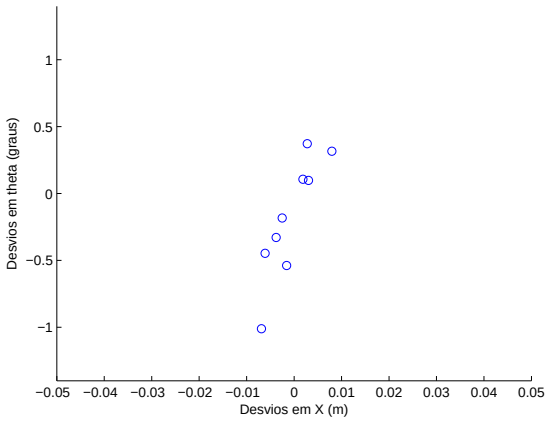
(c) Variação dos desvios de theta (graus) em relação aos desvios de Y (m)

Figura 6.9: Desvios observados em x e y e orientação utilizando o algoritmo detetor de linha para uma linha de 17.5 cm

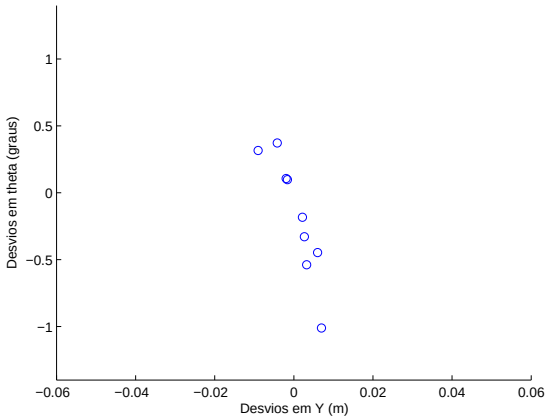
Na figura 6.9a é possível verificar que existe uma correlação inversamente proporcional entre os desvios em X e os desvios em Y. Nas figuras 6.9b e 6.9c, os desvios não se correlacionam.



(a) Variação dos desvios de Y (m) em relação aos desvios de X (m)



(b) Variação dos desvios de theta (graus) em relação aos desvios de X (m)



(c) Variação dos desvios de theta (graus) em relação aos desvios de Y (m)

Figura 6.10: Desvios observados em x e y e orientação utilizando o algoritmo detetor de linha para uma linha de 48 cm

Observando a figura 6.10a, é possível constatar que existe uma correlação inversamente proporcional entre o desvio no eixo X e o desvio em Y. Como é possível observar na figura 6.10b, com o aumento do desvio em X, há um aumento do desvio em θ no mesmo sentido. Na última figura (Figura 6.10c), os desvios correlacionam-se de forma inversamente proporcional entre si.

Como seria de esperar, a chapa com maior comprimento obteve melhores resultados, uma vez que mais pontos do laser foram utilizados para detetar a forma pretendida, o que por sua vez diminuiu a influência de *outliers* detectados. De facto, a redução do tamanho de chapa para um terço, provocou um aumento nos erros em, aproximadamente, 3 vezes. No entanto, utilizando uma linha de 48 cm, os desvios obtidos foram duas vezes superiores ao erros obtidos na tabela 6.1, o que permite afirmar que parte destes erros se devem ao método de deteção do marcador. Este resultados são nitidamente piores do que os obtidos com os *beacons*.

6.1.4 *Perfect Match*

Para algoritmo *Perfect Match*, foram realizados dois conjuntos de experiências: um primeiro onde se utilizou o algoritmo com as modificações descritas na Secção 5.3.1 e outro sem essas modificações. No entanto, foi necessário, para esta situação, calcular o peso que se pretende dar às medidas anteriores (α) no filtro passa-baixo. Sabendo que o robô móvel percorre uma trajetória linear de 2.4 metros de comprimento a uma velocidade de 0.15 m/s, este demorará 16 segundos para concluir a trajetória. Desta forma, através das equações (5.13) e (5.15), o pólo da função de transferência em tempo contínuo será:

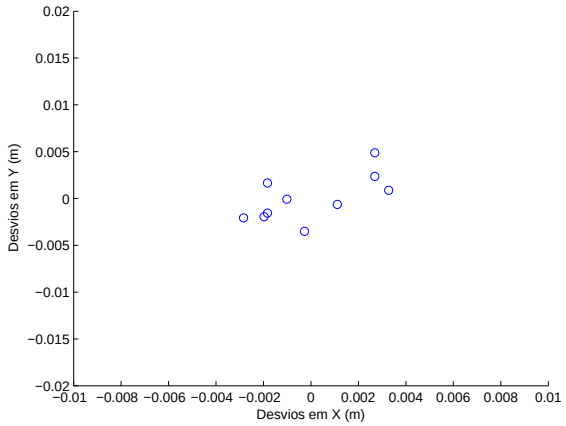
$$\sigma = -\frac{1}{\tau} = -\frac{1}{2}\frac{1}{segundos} \tag{6.4}$$

Passando para tempo discreto, através da função (5.16), o peso atribuído às medidas anteriores será de 0.9512.

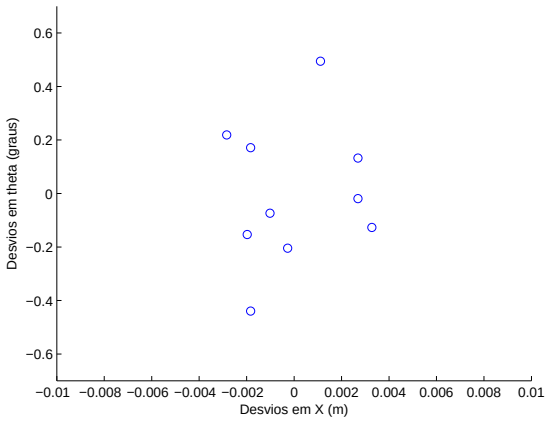
Após vários testes, foram obtidos os resultados presentes na tabela 6.4 e nas figuras 6.11 e 6.12.

Tabela 6.4: Média dos erros absolutos e desvio padrão das poses finais utilizando o algoritmo *Perfect Match* com e sem filtro

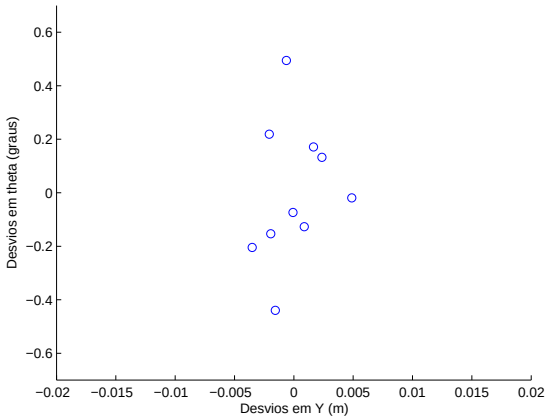
	μ_{erro}		σ	
	Sem Filtro	**Com filtro**	**Sem Filtro**	**Com Filtro**
x(m)	0.0020	0.0009	0.0022	0.0011
y(m)	0.0020	0.0015	0.0024	0.0019
θ**(rad)**	0.0036	0.0024	0.0043	0.0028
$\theta(^{\mathbf{o}})$	0.203	0.139	0.245	0.164



(a) Variação dos desvios de Y (m) em relação aos desvios de X (m)



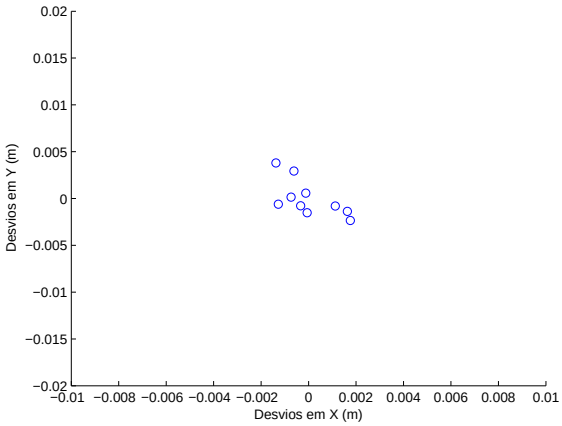
(b) Variação dos desvios de theta (graus) em relação aos desvios de X (m)



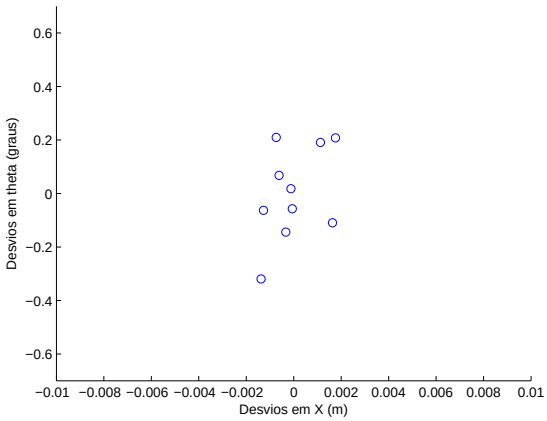
(c) Variação dos desvios de theta (graus) em relação aos desvios de Y (m)

Figura 6.11: Desvios observados em x e y e orientação utilizando o algoritmo *Perfect Match* (sem filtro)

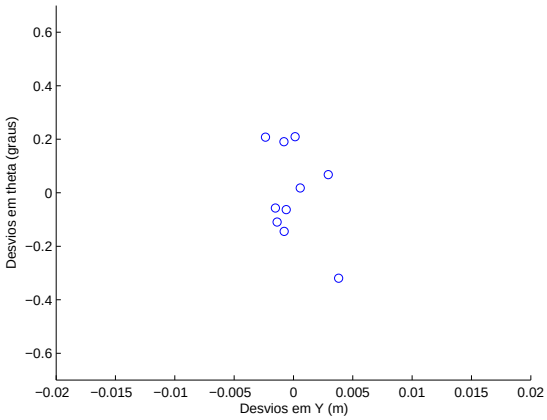
Na figura 6.11a é possível verificar que existe uma correlação diretamente proporcional entre os desvios em X e os desvios em Y. Nas figuras 6.11b e 6.11c, os desvios não se correlacionam.



(a) Variação dos desvios de Y (m) em relação aos desvios de X (m)



(b) Variação dos desvios de theta (graus) em relação aos desvios de X (m)



(c) Variação dos desvios de theta (graus) em relação aos desvios de Y (m)

Figura 6.12: Desvios observados em x e y e orientação utilizando o algoritmo *Perfect Match* (com filtro)

Na figura 6.12a é possível verificar que existe uma correlação inversamente proporcional entre os desvios em X e os desvios em Y. Nas figuras 6.12b e 6.12c, os desvios não se correlacionam.

Em relação aos dados obtidos, não é possível tirar nenhuma conclusão em concreto, visto que os erros se encontram na ordem de grandeza dos erros do sistema de localização e controlador. No entanto podemos afirmar que os erros são da mesma ordem de grandeza do método de localização de *Beacons* e que o filtro melhorou ligeiramente os resultados.

Após o término desta fase de testes, os dados obtidos foram analisados. Desta análise, verificou-se que os resultados utilizando o detetor de linha foram piores que os obtidos com os outros dois algoritmos, como é possível verificar nas tabelas e figuras anteriores. Desta forma, decidiu-se testar os dois algoritmos restantes no robô *Edgar* (Secção 3.5.1).

6.2 Resultados obtidos com o Edgar

Nesta fase de testes, será avaliada a precisão do sistema, sendo que o robô executará uma trajetória idêntica à dos testes anteriores. De facto, este iniciará de um ponto aleatório de onde detetará a *docking station*. Terminada esta fase, o robô móvel dirige-se para um ponto inicial longe da doca e orientado para esta. Atingida esta pose, o *Edgar* percorrerá uma linha de forma a aproximar-se do objeto de interesse, parando num ponto final. O comprimento do segmento de reta e a distância entre o ponto inicial e a *docking station* serão determinados através da informação obtida após a afinação do controlador seguidor de linha (Secção 4.1.1).

Observando a figura 4.10, verifica-se que o robô móvel demora, aproximadamente, 2.1 segundos a atingir o regime de estabilidade. No entanto, para cálculos futuros, considerou-se esse tempo como 3.6 segundos, por questões de segurança. Admitindo a velocidade linear deste como constante e de 0.15 m/s, é possível determinar o comprimento da linha (d).

$$v = \frac{d}{\Delta t} \Leftrightarrow d = 0,15 \cdot 3,6 = 0,54\,metros \tag{6.5}$$

De forma a assegurar que o robô atinja a posição final com um erro baixo, decidiu-se aumentar esta distância para, aproximadamente, o dobro, ou seja, 1 metro.

Ao contrário do *Jarvis*, o *Edgar* não possuí um laser de navegação, logo a pose final foi obtida manualmente, através de um ponteiro adicionado à frente do robô. Antes da inicialização do conjunto de testes, conduziu-se o robô móvel até à posição 0 (posição onde os ganchos se encontram alinhados com o centro das argolas) e guardou-se esta pose, como é possível observar na figura 6.13.

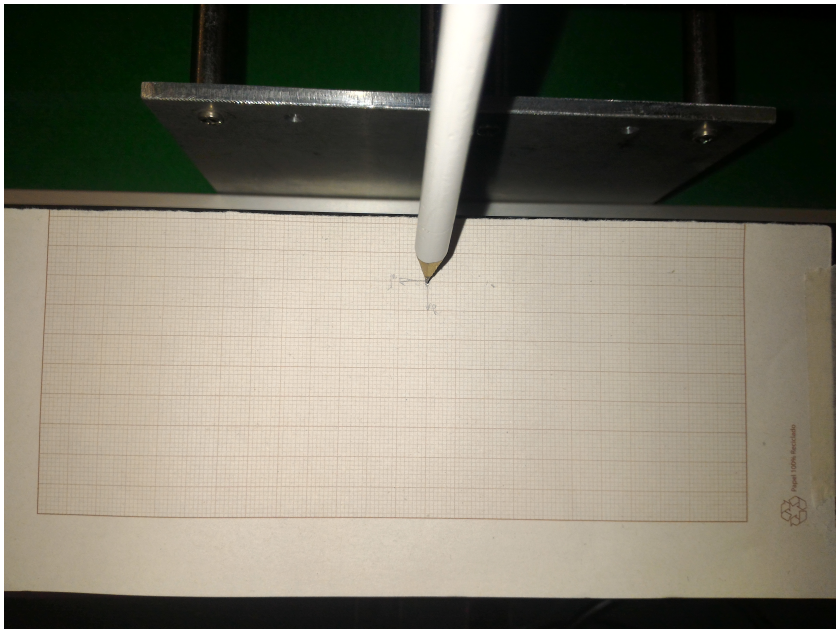


Figura 6.13: Exemplo de uma posição 0.

Esta pose será o ponto final da trajetória que o *Edgar* terá de percorrer, sendo que o ponto inicial estará 1.5 metro afastado da *docking station* (Figura 6.14). Devido às dimensões do robô, a posição final da trajetória ficará a 0.5 metros do objeto.

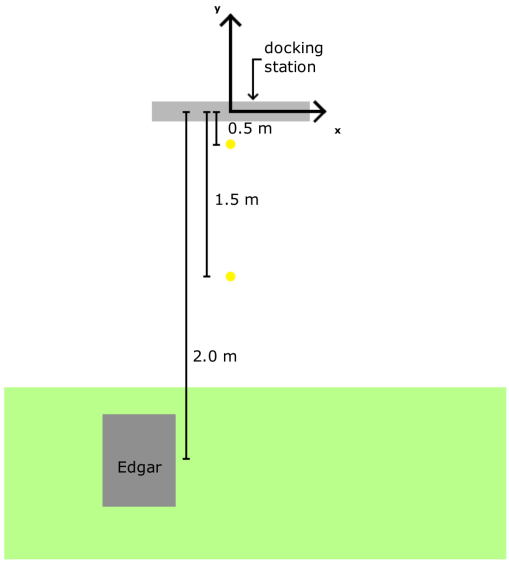


Figura 6.14: Exemplo da experiência efetuada com o Edgar. A área a verde corresponde a posições possíveis das quais o robô poderá ser inicializado

Nesta fase, foram testados os algoritmos de localização de *beacons* e *Perfect Match* com e sem modificações face a uma linha, sendo que após cada experiência será registado o desvio obtido em relação à posição 0. Calculou-se a média dos erros obtidos através da equação 6.6.

$$\mu_{desvio} = \frac{1}{N}\sum_{i=1}^{N} d_i \tag{6.6}$$

onde d_i representa os desvios obtidos na experiência i (d_x, d_y) e N o número de experiências efetuadas (neste caso 10).

Para avaliação dos resultados, será determinada a média dos erros absolutos em relação à média dos desvios (Equação 6.7), assim como o desvio padrão em relação a essa média (Equação 6.8).

$$\mu_{erro} = \frac{1}{N}\sum_{i=1}^{N} |\mu_{desvio} - d_i| \tag{6.7}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (\mu_{desvio} - d_i)^2} \tag{6.8}$$

onde μ_{desvio} representa a média dos desvios obtidos, d_i os desvios obtidos na experiência i (d_x, d_y) e N o número de experiências efetuadas.

6.2.1 Utilizando a *docking station*

Nesta fase decidiu-se testar os algoritmos de localização de *Beacons* e *Perfect Match* face à *docking station* utilizada anteriormente (Figura 6.5).

6.2.1.1 Com o algoritmo de localização de *Beacons*

Começou-se por testar o algoritmo de localização de *beacons* com e sem as modificações efetuadas.

Após o conjunto de testes, foram obtidos os resultados presentes na tabela 6.5 e nas figuras 6.15 e 6.16.

Tabela 6.5: Média dos erros absolutos, desvio padrão e desvio máximo obtidos utilizando o algoritmo de localização de *beacons* com e sem modificações

	μ_{erro}		σ		Desvio Máximo	
	Sem Modificações	Com Modificações	Sem Modificações	Com Modificações	Sem Modificações	Com Modificações
x(m)	0.005	0.002	0.007	0.003	0.012	0.006
y(m)	0.001	0.001	0.001	0.001	0.003	0.002

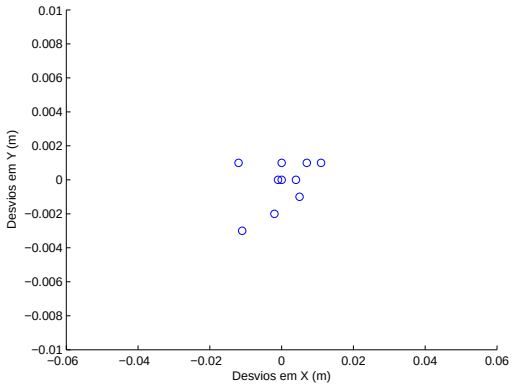


Figura 6.15: Desvios observados em x e y utilizando o algoritmo de localização de *beacons* sem modificações

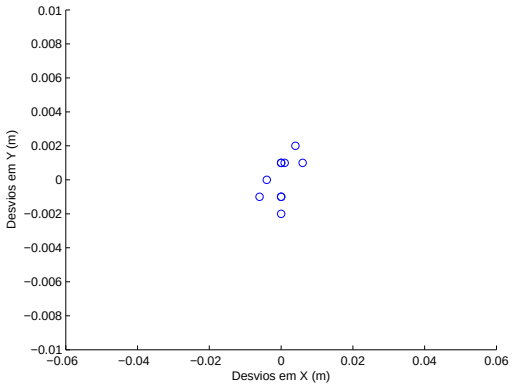


Figura 6.16: Desvios observados em x e y utilizando o algoritmo de localização de *beacons* com modificações

Como é possível observar através da tabela e figuras anteriores, após as modificações efetuadas ao algoritmo de localização de *beacons*, os resultados apresentaram uma dispersão menor do que os obtidos sem estas modificações. O algoritmo após as alterações apresentou também um desvio máximo menor do que o algoritmo sem alterações.

6.2.1.2 Com o *Perfect Match*

Em relação ao algoritmo *Perfect Match*, foi necessário recalcular o peso das medidas anteriores (α) no filtro passa-baixo. Sabendo que a velocidade linear se manteve a mesma (0.15 m/s) e que a distância a percorrer será de 1 metro, o robô demorará

$$\Delta t = \frac{1}{0.15} = \frac{20}{3}\ segundos \tag{6.9}$$

Desta forma, através da equação 5.13

$$\tau = \frac{1}{4} \cdot \frac{20}{3 \cdot 2} = \frac{10}{12}\ segundos \tag{6.10}$$

Através da equação 5.15, é possível calcular o pólo da função de transferência

$$\sigma = -\frac{1}{\tau} = -\frac{12}{10}\ \frac{1}{segundos} \tag{6.11}$$

Discretizando o pólo, através da equação 5.16, α será de 0.8869.

Após o conjunto de testes, foram obtidos os resultados presentes na tabela 6.6 e nas figuras 6.17 e 6.18 são apresentados os dados obtidos.

Tabela 6.6: Média dos erros absolutos, desvio padrão e desvio máximo obtidos utilizando o algoritmo *Perfect Match* com e sem filtro.

	μ_{erro}		σ		Desvio Máximo	
	Sem Filtro	Com Filtro	Sem Filtro	Com Filtro	Sem Filtro	Com Filtro
x(m)	0.017	0.007	0.020	0.008	0.033	0.014
y(m)	0.002	0.001	0.003	0.002	0.005	0.008

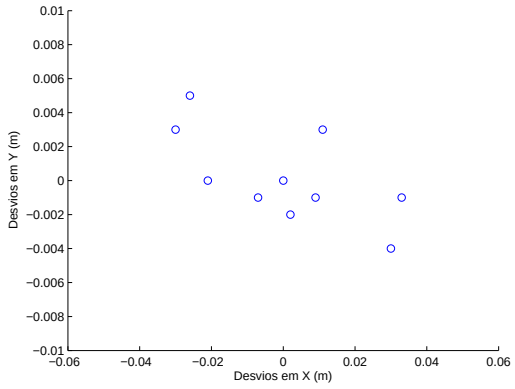


Figura 6.17: Desvios observados em x e y utilizando o algoritmo *Perfect Match* sem filtro

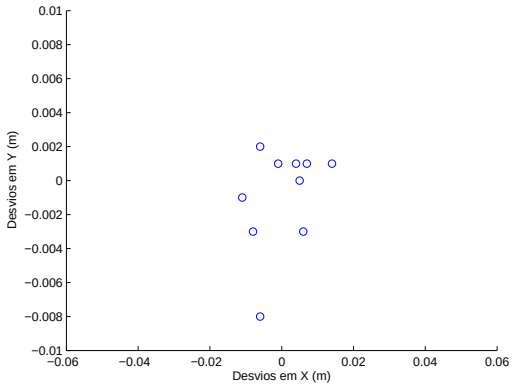


Figura 6.18: Desvios observados em x e y utilizando o algoritmo *Perfect Match* com filtro

Como é possível observar na tabela e nas figuras, a aplicação do filtro à resposta do algoritmo *Perfect Match*, permitiu diminuir a dispersão dos desvios da pose final, aumentando a precisão. No

entanto, a ordem de grandeza destes desvios é superior à ordem de grandeza dos erros permitidos (Secção 3.1). De facto, os erros obtidos em X ultrapassam o desvio máximo transversal de ± 12 milímetros, desta forma, decidiu-se testar com uma filtragem que dê-se uma maior peso às medidas anteriores, admitindo que a resposta necessitará de mais tempo para estabilizar, desse modo, a equação 5.13 passa a ser

$$\tau = \frac{1}{4} \cdot \frac{3 \cdot \Delta t}{4} \; segundos \tag{6.12}$$

Através da equação 5.16, o peso atribuído às medidas anteriores (α) será de 0.9231.

Voltou-se a fazer um conjunto de testes, agora com este filtro, obtendo-se os resultados expostos na tabela 6.7 e na figura 6.19.

Tabela 6.7: Média dos erros absolutos, desvio padrão e desvio máximo obtidos utilizando o algoritmo *Perfect Match* com novo filtro.

	μ_{erro}	σ	Desvio Máximo
x(m)	0.004	0.005	0.011
y(m)	0.001	0.001	0.002

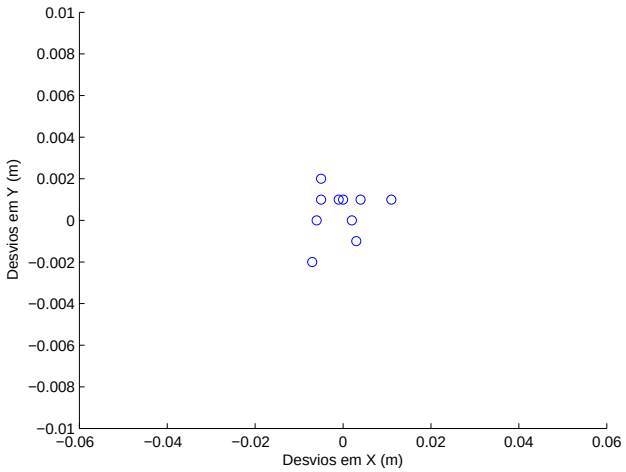


Figura 6.19: Desvios observados em x e y utilizando o algoritmo *Perfect Match* com novo filtro.

Como é possível observar, os erros foram menores do que utilizando o filtro anterior, sendo que se apresentam dentro da ordem de grandeza dos erros máximos permitidos.

Analisando as tabelas e as figuras anteriores, os algoritmos com as modificações efetuadas apresentaram melhores resultados do que as versões originais dos mesmos. De facto, apresentaram uma menor dispersão dos desvios observados, sendo que estes foram inferiores aos erros máximos permitidos.

6.2.2 *Perfect Match* utilizando outras formas

Após os resultados obtidos utilizando uma linha, decidiram-se testar outras formas, como é possível observar nas figuras 6.20 e 6.21.

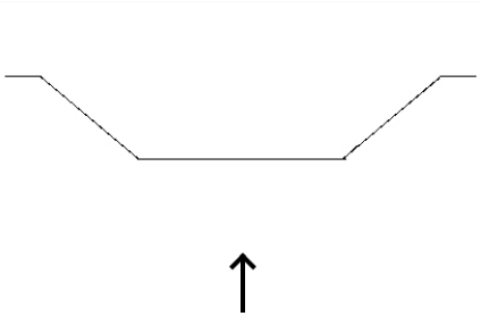


Figura 6.20: Forma em V. A seta representa a direção da aproximação do robô.

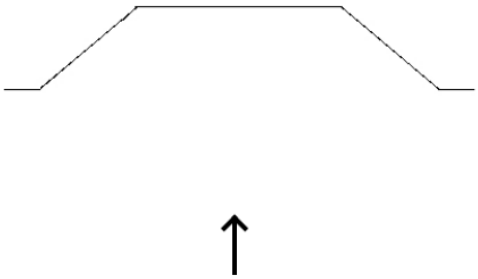


Figura 6.21: Forma em V invertido. A seta representa a direção da aproximação do robô.

Ambas foram obtidas utilizando uma única chapa com as medidas expostas na figura 6.22.

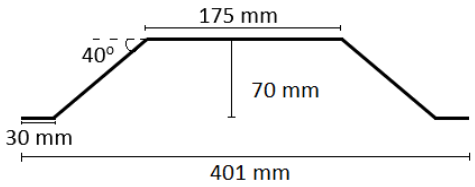


Figura 6.22: Chapa com medidas.

Para esta fase de testes, decidiu-se testar o algoritmo *Perfect Match*, mas só com as modificações efetuadas.

Após um conjunto de 10 testes, foram obtidos os seguintes resultados expostos na tabela 6.8 e nas figuras 6.23 e 6.24.

Tabela 6.8: Média dos erros absolutos, desvio padrão e desvio máximo obtidos utilizando o algoritmo *Perfect Match* com formas diferentes.

	μ_{erro}		σ		Desvio Máximo	
	Chapa em V	Chapa em V invertido	Chapa em V	Chapa em V invertido	Chapa em V	Chapa em V invertido
x(m)	0.003	0.003	0.004	0.003	0.006	0.006
y(m)	0.001	0.001	0.001	0.001	0.002	0.002

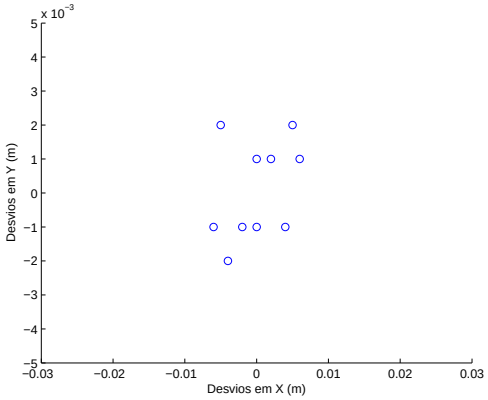


Figura 6.23: Desvios observados em x e y utilizando o algoritmo *Perfect Match* com a chapa normal

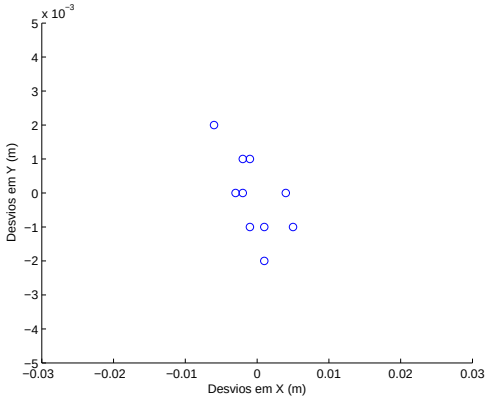


Figura 6.24: Desvios observados em x e y em relação à posição 0 utilizando o algoritmo *Perfect Match* com a chapa invertida

Analisando a tabela e as figuras anteriores, não é possível tirar nenhuma conclusão, pois os erros obtidos foram da mesma ordem de grandeza, no entanto, estes desvios apresentam-se dentro da gama máxima permitida (Secção 3.1). Comparados estes resultados com os obtidos utilizando uma linha (Tabela 6.7), é possível observar que houve melhorias. O desvio máximo obtido foi menor, assim como a dispersão de pontos.

Como a chapa utilizada para esta fase de testes tem um comprimento menor do que a *docking station* utilizada na fase anterior, decidiu-se então testar o algoritmo de localização de *beacons* nas dimensões da chapa, de modo a obter-se uma comparação mais justa.

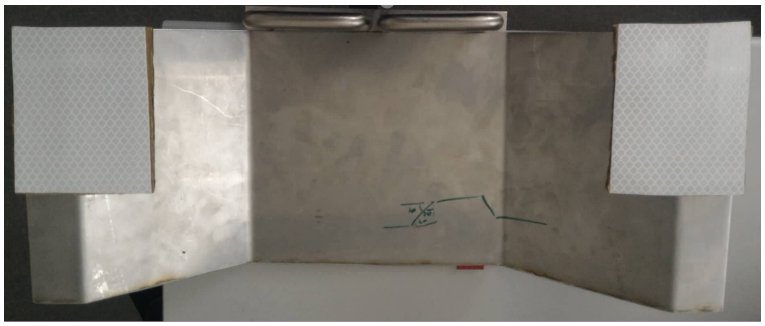


Figura 6.25: *Beacons* colocados na chapa.

Após um conjunto de 10 testes, foram obtidos os seguintes resultados expostos na tabela 6.9 e na figura 6.26.

Tabela 6.9: Média dos erros absolutos, desvio padrão e desvio máximo obtidos utilizando o algoritmo de localização de *beacons*.

	μ_{erro}	σ	Desvio Máximo
x(m)	0.003	0.003	0.007
y(m)	0.001	0.001	0.002

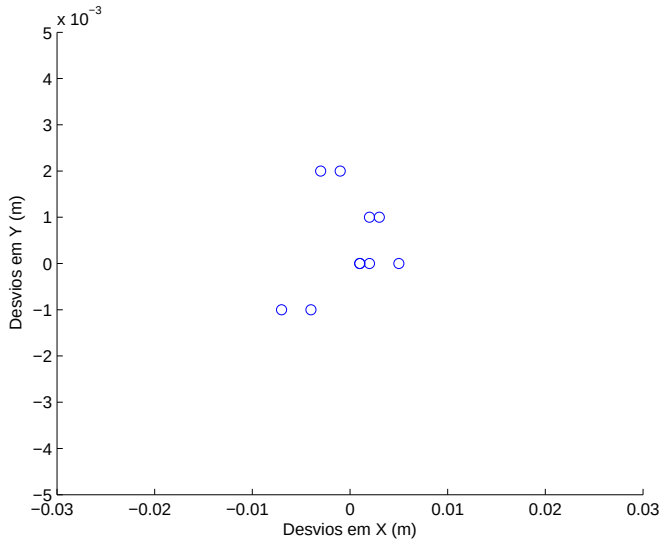


Figura 6.26: Desvios observados em x e y utilizando o algoritmo de localização de *beacons*.

Observando a tabela e a figura anterior e comparando com os resultados obtidos utilizando o *Perfect Match* com a forma em V ou em V invertido, a ordem de grandeza dos desvios de ambos os métodos é idêntica, não permitindo tirar conclusões sobre qual o melhor.

Comparando estas formas com a linha, o algoritmo *Perfect Match* apresentou uma ligeira melhoria em termos de precisão, quando utilizada uma forma mais complexa. Relativamente ao algoritmo de localização de *beacons*, os erros obtidos nas duas situações são da mesma ordem de grandeza. De facto, aproximando os marcos, não houve uma variação significativa dos desvios apresentados.

Quanto aos marcos utilizados nesta fase de testes (beacons e chapas), a colocação dos primeiros é mais simples do que a construção e fixação das segundas. Desta forma, pode-se aconselhar a utilização do algoritmo de localização de Beacons. No entanto, o algoritmo Perfect Match pode ser utilizado caso o objeto onde se pretende atracar já possua uma forma semelhante àquela utilizada.

Capítulo 7

Conclusão e Trabalho futuro

O projeto da presente dissertação focou-se na implementação de um sistema de atracagem preciso. Este é composto por dois módulos: um responsável pela deteção do marco e outro encarregue da aproximação ao objeto de interesse.

Numa primeira fase, o robô *Jarvis* foi levado a percorrer uma reta num referencial fixo, sendo observadas as poses finais. Tal permitiu testar o sistema de localização, navegação e o controlador deste robô, ficando-se assim com uma ideia do ruído/desvios associados aos mesmos.

Com estes desvios em mente, foram avaliados os algoritmos de deteção de marcos perante uma linha, tendo-se observado também a repetibilidade da pose final do robô. Verificou-se que: quer utilizando o algoritmo de localização de *beacons* quer o *Perfect Match*, ambos com e sem as modificações efetuadas, os erros obtidos foram da mesma ordem de grandeza que os desvios verificados pelo sistema de navegação e controlador do robô *Jarvis*. O mesmo não aconteceu utilizando o algoritmo detetor de linha, de facto este apresentou erros maiores. Desta forma, este método foi descartado para a segunda fase de testes.

Numa segunda fase, os algoritmos restantes foram testados no robô *Edgar* face à mesma linha. Antes da inicialização destes testes, o controlador de trajetória foi afinado, obtendo-se erros finais absolutos de 1 milímetro. Comparando os diferentes métodos, observou-se que as modificações efetuadas aos algoritmos de localização de *beacons* e *Perfect Match* foram bem sucedidas, tendo os desvios finais apresentado uma dispersão menor, bem como um desvio máximo menor do que os métodos sem alterações.

Numa terceira fase, o algoritmo *Perfect Match* com modificações foi testado, utilizando duas formas mais complexas. Comparando os resultados obtidos com os da linha, observou-se uma ligeira melhoria dos resultados finais. Como estas formas tinham um comprimento menor do que a linha, foi testado o algoritmo de localização de *beacons* com estes colados à chapa. Tal serviu para verificar se uma redução da distância entre marcos influenciaria a resposta final. Após o conjunto de experiências, verificou-se que os resultados foram semelhantes aos anteriores.

Comparando os *beacons* com as chapas utilizadas, os primeiros apresentam algumas vantagens, nomeadamente na colocação destes. De facto, a construção e fixação das chapas torna-se um processo mais demorado e complicado comparado com a fixação de refletores. No entanto, a

utilização do algoritmo *Perfect Match* pode ser considerada caso a *docking station* apresente uma forma semelhante à das chapas utilizadas.

Concluindo, foi possível desenvolver um sistema de atracagem com precisão. Os erros obtidos apresentam-se na mesma ordem de grandeza dos requisitos pedidos (Secção 3.1).

7.1 Trabalho futuro

Após a implementação do sistema de atracagem, certos melhoramentos podem ser efetuados de forma a aperfeiçoar o seu desempenho.

No módulo do sistema de deteção da *docking station*, certas modificações podem ser levadas a cabo de modo a melhorar a resposta daquele. Em relação ao algoritmo detetor de linha, um filtro de Kalman que estimasse a posição do robô em relação ao marco, poderia ser desenvolvido com o objetivo de filtrar a resposta deste método. Este também poderia ser modificado com a finalidade de possibilitar a deteção de formas mais complexas, em vez de uma única linha. O detetor de localização de *beacons* poderia ser alterado de forma a ficar mais robusto a *outliers*. De facto, caso existisse um refletor indesejado na zona de interesse, este poderia ser eliminado de modo a diminuir o seu impacto no sistema. Para o algoritmo *Perfect Match*, como trabalho futuro pode ser explorada a afinação do filtro de Kalman, de modo a eliminar a necessidade do filtro passa-baixo.

Ao nível do módulo de aproximação à *docking station*, é possível aperfeiçoar o modo como o robô móvel executa a trajetória. De facto, em vez de executar dois movimentos - dirigir-se para um ponto e depois percorrer um segmento de reta, poderia ser efetuado uma única trajetória suave em direção à posição final. Em relação ao controlador seguidor de linha, como melhoria poderia ser implementada uma componente derivativa neste, de modo a obter-se uma resposta mais rápida.

Referências

[1] W. Wang, Z. Li, W. Yu, e J. Zhang. An autonomous docking method based on ultrasonic sensors for self-reconfigurable mobile robot. Em *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, páginas 1744–1749.

[2] Li Dazhai, Fu Hualei, e Wang Wei. Ultrasonic based autonomous docking on plane for mobile robot. Em *2008 IEEE International Conference on Automation and Logistics*, páginas 1396–1401.

[3] R. C. Luo, C. H. Huang, e C. Y. Huang. Search and track power charge docking station based on sound source for autonomous mobile robot applications. Em *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, páginas 1347–1352.

[4] R. C. Luo, C. T. Liao, e S. C. Lin. Multi-sensor fusion for reduced uncertainty in autonomous mobile robot docking and recharging. Em *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, páginas 2203–2208.

[5] M. Kim, H. W. Kim, e N. Y. Chong. Automated robot docking using direction sensing rfid. Em *Proceedings 2007 IEEE International Conference on Robotics and Automation*, páginas 4588–4593.

[6] D. Amarasinghe, G. K. I. Mann, e R. G. Gosine. Vision-based hybrid control strategy for autonomous docking of a mobile robot. Em *Proceedings of 2005 IEEE Conference on Control Applications, 2005. CCA 2005.*, páginas 1600–1605.

[7] Zhang Zhigao, Qu ZhenShen, e Li Qinghua. An autonomous docking simulation system based on monocular vision. Em *2006 1st International Symposium on Systems and Control in Aerospace and Astronautics*, páginas 4 pp.–1315.

[8] R. C. Luo, C. T. Liao, K. L. Su, e K. C. Lin. Automatic docking and recharging system for autonomous security robot. Em *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, páginas 2953–2958.

[9] R. C. Luo, C. T. Liao, e K. C. Lin. Vision-based docking for automatic security robot power recharging. Em *IEEE Workshop on Advanced Robotics and its Social Impacts, 2005.*, páginas 214–219.

[10] U. Kartoun, H. Stern, Y. Edan, C. Feied, J. Handler, M. Smith, e M. Gillam. Vision-based autonomous robot self-docking and recharging. Em *2006 World Automation Congress*, páginas 1–8.

[11] Martin Lauer, Sascha Lange, e Martin Riedmiller. *Calculating the Perfect Match: An Efficient and Accurate Approach for Robot Self-localization*, páginas 142–153. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[12] Héber Sobreira, Miguel Pinto, António Paulo Moreira, Paulo Gomes Costa, e José Lima. *Robust Robot Localization Based on the Perfect Match Algorithm*, páginas 607–616. CONTROLO'2014 – Proceedings of the 11th Portuguese Conference on Automatic Control. Lecture Notes in Electrical Engineering, vol 321. Springer International Publishing, Cham, 2015.

[13] N. Jain, Y. P. Kumar, e K. S. Nagla. Corner extraction from indoor environment for mobile robot mapping. Em *2015 Annual IEEE India Conference (INDICON)*, páginas 1–6.

[14] B. X. Hon, H. Tian, F. Wang, B. M. Chen, e T. H. Lee. A customized fastslam algorithm using scanning laser range finder in structured indoor environments. Em *2013 10th IEEE International Conference on Control and Automation (ICCA)*, páginas 640–645.

[15] F. Tan, J. Yang, J. Huang, W. Chen, W. Wang, e J. Wang. A corner and straight line matching localization method for family indoor monitor mobile robot. Em *The 2010 IEEE International Conference on Information and Automation*, páginas 1902–1907.

[16] T. Weerakoon, K. Ishii, e A. A. F. Nassiraei. Geometric feature extraction from 2D laser range data for mobile robot navigation. Em *2015 IEEE 10th International Conference on Industrial and Information Systems (ICIIS)*, páginas 326–331.

[17] V. Nguyen, A. Martinelli, N. Tomatis, e R. Siegwart. A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. Em *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, páginas 1929–1934.

[18] Li Xinzhao, Liu Yuehu, Niu Zhenning, e Cui Zhichao. A line segments extraction based undirected graph from 2D laser scans. *Em 2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP)*, páginas 1–6.

[19] A. R. Corregedor, J. Meyer, e F. Du Plessis. Design principles for 2D local mapping using a laser range finder. Em *AFRICON, 2011*, páginas 1–6.

[20] Qiu Quan e Han Jianda. An implementation of typical-obstacle detection and recognition with laser range finder. Em *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, volume 2, páginas 742–746.

[21] M. Norouzi, M. Yaghobi, M. R. Siboni, e M. Jadaliha. Recursive line extraction algorithm from 2D laser scanner applied to navigation a mobile robot. Em *2008 IEEE International Conference on Robotics and Biomimetics*, páginas 2127–2132.

[22] M. Liu, X. Lei, S. Zhang, e B. Mu. Natural landmark extraction in 2D laser data based on local curvature scale for mobile robot navigation. Em *2010 IEEE International Conference on Robotics and Biomimetics*, páginas 525–530.

[23] X. Feng, Y. He, W. Huang, e J. Yuan. Natural landmarks extraction method from range image for mobile robot. Em *2009 2nd International Congress on Image and Signal Processing*, páginas 1–5.

[24] P. Nunez, R. Vaizquez, J. C. del Toro, A. Bandera, e F. Sandoval. A curvature based method to extract natural landmarks for mobile robot navigation. Em *2007 IEEE International Symposium on Intelligent Signal Processing*, páginas 1–6.

[25] N. Certad, R. Acuna, Terrones Á, D. Ralev, J. Cappelletto, e J. C. Grieco. Study and Improvements in Landmarks Extraction in 2D Range Images Based on an Adaptive Curvature Estimation. Em *2012 VI Andean Region International Conference*, páginas 95–98.

[26] R. C. Luo, Lin Shih-Chi, e Lai Chun Chi. Indoor autonomous mobile robot localization using natural landmark. Em *2008 34th Annual Conference of IEEE Industrial Electronics*, páginas 1626–1631.

[27] Q. Xu, C. Ren, H. Yan, e J. Ji. Laser sensor based localization of mobile robot using unscented kalman filter. Em *2016 IEEE International Conference on Mechatronics and Automation*, páginas 1726–1731.

[28] F. Schwiegelshohn, P. Wehner, F. Werner, D. Gohringer, e M. Hubner. Enabling indoor object localization through bluetooth beacons on the radio robot platform. Em *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, páginas 328–333.

[29] J. Krejsa e S. Věchet. Odometry-free mobile robot localization using bearing only beacons. Em *Proceedings of 14th International Power Electronics and Motion Control Conference EPE-PEMC 2010*, páginas T5–40–T5–45.

[30] A. Lobo, R. Kadam, S. Shajahan, K. Malegam, K. Wagle, e S. Surve. Localization and tracking of indoor mobile robot with beacons and dead reckoning sensors. Em *Electrical, Electronics and Computer Science (SCEECS), 2014 IEEE Students' Conference on*, páginas 1–4.

[31] Lee Sooyong e Song Jae-Bok. Mobile robot localization using infrared light reflecting landmarks. Em *2007 International Conference on Control, Automation and Systems*, páginas 674–677.

[32] J. López, D. Pérez, E. Zalama, e J. Gómez-García-Bermejo. Low cost indoor mobile robot localization system. Em *2011 11th International Conference on Intelligent Systems Design and Applications*, páginas 1134–1139.

[33] Kuo-Lan Su Lin, Yi-Lin Liao, Shih-Ping Lin, e Sian-Fu. An interactive auto-recharging system for mobile robots. *International Journal of Automation and Smart Technology*, 4, 2014. URL: http://www.ausmt.org/index.php/AUSMT/article/view/197.

[34] K. L. Su, B. Y. Li, J. H. Guo, e Y. L. Liao. Motion planning of multi-docking system for intelligent mobile robots. Em *2014 International Symposium on Computer, Consumer and Control*, páginas 1279–1282.

[35] MIRA Project. 2017. "Domain DockingProcess - Documentation". Acedido a 22 de Junho de 2017. URL: http://www.mira-project.org/MIRA-doc/domains/navigation/DockingProcess/index.html#DockingStations.

[36] A. Paulo G. M. Moreira. Cinemática, dinâmica e controlo de robôs. página 70, 2013.

[37] Tore Hagglund Karl J. Astrom. *PID Controllers: Theory, Design and Tuning*. 2nd edição, 1995. URL: https://aiecp.files.wordpress.com/2012/07/1-0-1-k-j-astrom-pid-controllers-theory-design-and-tuning-2ed.pdf.

[38] J. L. Martins de Carvalho. *Dynamical systems and automatic control*. 1993.

[39] Héber Sobreira, A Paulo Moreira, Paulo Gomes Costa, e José Lima. Robust mobile robot localization based on security laser scanner. Em *Autonomous Robot Systems and Competitions (ICARSC), 2015 IEEE International Conference on*, páginas 162–167. IEEE.

[40] H. Sobreira. *Fiabilidade e robustez da localização de robôs móveis*. Thesis, 2017.

[41] H. Sobreira, L. Rocha, C. Costa, J. Lima, P. Costa, e A. P. Moreira. 2D Cloud Template Matching - A Comparison between Iterative Closest Point and Perfect Match. Em *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, páginas 53–59.